REPORT DOCUMENTATION PAGE	Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
25-08-2015	MS Thesis	-

4. TITLE AND SUBTITLE
Matrix Product State Simulations of Quantum Algorithms

5a. CONTRACT NUMBER
W911NF-13-1-0024

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHORS
Kieran Woolfe

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
University of New South Wales
Research Office
University of New South Wales
-

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
63162-PH-OC.101

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
We develop simulation methods for matrix product operators, and perform simulations of the Quantum Fourier Transform, Shor's algorithm and Grover's algorithm using matrix product states and matrix product operators. By doing so, we provide numerical evidence that a constant number of QFTs can be e♬ciently classically simulated on any state whose Schmidt rank grows only polynomially with the number of qubits, and quantify the amount of entanglement present in Shor's algorithm. The e♬ciency of the matrix product state and operator representation

15. SUBJECT TERMS
quantum computing, quantum algorithms, Quantum Fourier Transform, Shor's algorithm, Grover's algorithm, large-scale quantum algorithm simulations

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Andrew Dzurak
UU	UU	UU	UU		19b. TELEPHONE NUMBER
					612-938-5631

Standard Form 298 (Rev 8/98)
Prescribed by ANSI Std. Z39.18

Report Title

Matrix Product State Simulations of Quantum Algorithms

ABSTRACT

We develop simulation methods for matrix product operators, and perform simulations of the Quantum Fourier Transform, Shor's algorithm and Grover's algorithm using matrix product states and matrix product operators. By doing so, we provide numerical evidence that a constant number of QFTs can be eﬃciently classically simulated on any state whose Schmidt rank grows only polynomially with the number of qubits, and quantify the amount of entanglement present in Shor's algorithm. The eﬃciency of the matrix product state and operator representation allows us to perform moderately large simulations of both Shor's algorithm with Z errors and Grover's algorithm with up to 15 X, Y and Z errors. While larger simulations have been performed, our results have been computed with little computational power and provide new methods to perform large-scale quantum algorithm simulations.

Matrix Product Operator Simulations of Quantum Algorithms

Kieran Woolfe

Submitted in total fulfillment of the requirements of the degree of Master of Philosophy

School of Physics
The University of Melbourne

February, 2015

Abstract

We develop simulation methods for matrix product operators, and perform simulations of the Quantum Fourier Transform, Shor's algorithm and Grover's algorithm using matrix product states and matrix product operators. By doing so, we provide numerical evidence that a constant number of QFTs can be efficiently classically simulated on any state whose Schmidt rank grows only polynomially with the number of qubits, and quantify the amount of entanglement present in Shor's algorithm. The efficiency of the matrix product state and operator representation allows us to perform moderately large simulations of both Shor's algorithm with Z errors and Grover's algorithm with up to 15 X, Y and Z errors. While larger simulations have been performed, our results have been computed with little computational power and provide new methods to perform large-scale quantum algorithm simulations.

Declaration

This is to certify that:

i the thesis comprises only my original work towards the MPhil except where indicated in the Preface,

ii due acknowledgement has been made in the text to all other material used,

iii the thesis is less than 50 000 words in length, exclusive of tables, maps, bibliographies and appendices.

Kieran Woolfe
February, 2015

Preface

This thesis is comprised entirely of original work undertaken during this Masters of Philosophy under the supervision of Dr Charles Hill and Professor Lloyd Hollenberg, except for the reviews of earlier work and the literature as noted in the text. In particular, chapters 2 and 3 are reviews of existing theory relevant to the thesis and sections 4.1, 5.1 and 6.1 are reviews of the previous work relevant to the topics of chapters 4, 5 and 6 respectively. The majority of the content of chapter 4 is the same as that of a paper prepared for publication during the Masters program [1], which was authored with the guidance of Dr Hill and Professor Hollenberg. All simulations discussed in this thesis were performed with original computer codes developed for this project, using the LAPACK and NumPy libraries.

Acknowledgements

I extend my thanks to everyone who has assisted me and made this research possible. Firstly and most importantly to my supervisors Dr Charles Hill and Professor Lloyd Hollenberg for their extensive assistance throughout the duration of my research and the preparation of this document. I acknowledge the assistance of the Australian Government for the grant of an Australian Postgraduate Award which made my research possible. My thanks also to the everyone from the Australian Research Council Center of Excellence for Quantum Computation and Communication Technology for providing me with an intellectual home in which to learn about quantum computing and to collaborate. Particular thanks to Ms Patricia Gigliuto for making everything administrative very simple for me. I extend my gratitude to the School of Physics at the University of Melbourne for admitting me to their graduate program and to the libraries and staff of the university for facilitating my research. Finally, my thanks to my fellow graduate students and to my office-mates, who were instrumental in both delaying and hastening the completion of this document.

Contents

Chapter 1

Introduction

Quantum computing, the use of quantum devices to perform computations, may deliver exponentially faster results than classical computing for specific problems. Most famously, Shor's algorithm [2, 3] allows the products of primes to be factorised with an exponential speedup over the fastest known classical algorithm. However, a quantum computer would be subject to noise and may only be useful for selected problems which must be identified and solved with carefully designed algorithms. Research in the field of quantum algorithms offers ways to address these issues. In this thesis, we focus on two questions: what is the source of the computational speedup in quantum computing, and how will a quantum computer perform in the presence of noise. We address these problems with simulations of quantum algorithms.

Quantum Computing

The idea of quantum computing stems from the exponential growth of the size of the Hilbert space of a quantum system as the system size increases. The very large size of such a Hilbert space both makes simulating large scale quantum mechanical systems very difficult and opens the possibility that controllable systems could be employed as computers to simulate other interesting quantum systems, an idea was first suggested by Feynman [4]. More broadly, quantum computing exploits the computational resources of large Hilbert spaces to perform calculations. Some of the quantum algorithms that have been discovered are: Shor's algorithm [2, 3], by which products of large primes are factorised; Grover's algorithm [5, 6], by which an unordered database can be searched with a quadratic speedup over classical methods;

and quantum simulation [7, 8, 9, 10, 11] by which it is believed that all physically realistic Hamiltonians can be simulated in polynomial time and the ground and thermal states of Hamiltonians can be estimated [12]. Quantum walks [13, 14], a quantum analogy of classical random walks, have been used to create quantum algorithms for such problems as search algorithms [15, 16, 17], the triangle problem [18] and verification of matrix products [19]. Quantum algorithms have also been described to solve a variety of algebraic problems such as hidden subgroup problems [20, 21, 22, 23, 24], hidden shift problems [25, 26], and hidden nonlinear structure problems [27]. Finally, while in traditional quantum computation, computations are carried out by applying quantum operations to a system consisting of pure qubits (two level quantum systems), alternative models exist such as adiabatic quantum computing [28, 29, 30, 31], in which the solution to a relevant problem is encoded in the ground state of a Hamiltonian and the Hamiltonian of a system in a simple ground state is evolved slowly to the more complicated Hamiltonian; and the one clean qubit model of quantum computation [32] in which computations are carried out with all qubits except one in a mixed state. Several more complete surveys and reviews of quantum algorithms have been made [33, 34, 35, 36, 37].

The fact that quantum mechanical systems can perform some computations exponentially faster than the fastest known classical systems invites the question of the origin of this computational speedup. The speedup must be caused by features of quantum mechanics that are inherently quantum mechanical and so are not reproducible in classical systems. Quantum entanglement is an natural candidate for such an effect, but it is still unclear to what degree it is responsible for, or even necessary for, a speedup [38, 39, 40, 41, 42, 43, 44, 45, 46]. Some quantum processes have been found [47, 48, 49, 50] which exhibit some computational speedup over classical methods, but display fixed entanglement as the size of the quantum system increases. Besides entanglement, another measurement of quantum correlations is discord [51, 52], and there are suggestions that it may be required in order for a speedup to occur [53, 54, 55]. Simulations of quantum computers are needed in order to quantify the level of entanglement and other correlations such as discord during these algorithms. In this thesis, we quantify the entangelement present during simulations of Shor's and Grover's algorithms.

Another method of studying the origin of quantum computing's computational speedup is to study which quantum mechanical procedures can be efficiently classically simulated. By developing new classical simulation techniques, parts of algorithms may be identified which permit efficient classical simulation. This identification limits the parts of quantum algorithm from which an advantage can and cannot arise over classical computing. Any quantum mechanical algorithm, or part of an algorithm, which permits efficient classical simulation cannot lead to a speedup during quantum processing. We perform simulations related to the source of quantum computational speedup in this thesis using the matrix product state and matrix product operator representations of quantum states and operators.

A physical quantum computer would be constantly interacting with its environment and would be difficult to control to the very high level of precision demanded to perform quantum computational tasks. As such, the computer would undergo decoherence and would not perform precisely the gates and algorithms the experimenter would require. A variety of quantum error correction codes have been developed to deal with this difficulty and to make quantum computers robust [56]. Among the most promising of these is the surface code [57]. A variety of results exist for the robustness of this code depending upon the assumptions made, with some indicating that the code could be robust to noise levels of up to 1% [58, 59]. This high error tolerance comes at the cost of using large numbers of qubits, with an estimate for the number of qubits need to factor a 2000 bit number of 20×10^7 [57]. This number of qubits is well beyond current capabilities, and so it is interesting to ask what computations could be performed on medium sized quantum computers in the presence of a small but non-zero number of errors. In this thesis we develop simulations that operate in a suitable regime to study this question. With the simulation of noisy quantum computations we also seek to shed light on the fundamental nature and operation of the algorithms we simulate. For this reason, we perform simulations of both Shor's algorithm and Grover's algorithm and quantify the probability of success of these algorithms under various error models.

Simulating quantum algorithms

We perform simulations related to the source of quantum computational speedups in this thesis using the matrix product state and matrix product operator representations of quantum states and operators. Matrix product states first appeared in the density matrix renormalisation group method [60, 61] (DMRG). This is a numerical method of determining the ground state of quantum mechanical systems. It achieved high levels of accuracy and use in one-dimensional systems. It was later determined that DMRG operates on matrix product states [62, 63], which naturally encode states in a way that is transparently related to the amount of entanglement they possess. Matrix product states allow small scale entanglement to be easily truncated from a system, leaving only the most important correlations. Separately to DMRG results, they have been separately shown to be able to simulate lightly entangled quantum systems efficiently [40]. As such, they may be used to efficiently simulate selected large-scale quantum algorithms.

Matrix product states encode the information of a quantum state in local manner by a one dimensional network of tensors. There are several generalisations of matrix product state. These include projected entangled pair states [64], which encode quantum states in higher numbers of spatial dimensions, but can be more difficult to efficiently simulate than matrix product states. Matrix product operators [65, 66] extend the matrix product state formalism to describe quantum mechanical *operators* in a one dimensional tensor network. These operators are used to perform time evolution on matrix product states [67, 68] in condensed matter calculations. However, they have not been extensively used in the past to study the operation of quantum algorithms. In this thesis we use matrix product operators as well as matrix product states in the simulation of quantum algorithms. These states and operators allow quantum information to be presented in a highly local format and so allow the simulation of larger systems than many other simulation methods. As a result, we are able to simulate large instances of Grover's and Shor's algorithm with simple error models.

Several other simulation results exist for the efficient classical simulability of quantum algorithms. The Gottesman-Knill theorem [69, 70] states that circuits composed only of quantum gates from a particular group, the Clifford group, can be efficiently simulated. These techniques have been extended to show the simulability of quantum Fourier transforms and normalizer circuits over Abelian groups [71, 72]. The simulation of a different group of gates which called match-gates [73, 74] can be shown to be equivalent to the simulation of fermionic linear optics[75, 76], which is efficiently classically simulable. Vidal has shown that quantum circuits that generate limited amounts of entanglement are efficient classically simulable by using matrix product states [40]. Another approach is to view quantum circuits as graphs and to simulate them by contracting indices one by one. Considering circuits this way, it has been shown that any circuit whose graph representation has restricted topological properties, such as a small tree-width, may be efficiently classically simulated [41, 77, 78]. Finally, the efficient simulation of any circuit with a sparse output distribution has been shown [79]. Our work computationally builds on this earlier work with matrix product simulations to obtain further results about the simulability of the quantum Fourier transform and Grover's algorithm.

Outline

In chapters 1 and 2 of this thesis we provide introductions to quantum computing and matrix product states and operators respectively. In chapter 2 we also detail our methods of quantum circuit simulation including original work on how to add matrix product operators during quantum algorithm simulation. In chapter 3 we form the matrix product operator representation of the quantum Fourier transform and provide computational evidence that a constant number of transforms can be efficiently classically simulated for all states with Schmidt ranks which grow only polynomially with the number of qubits. We use our quantum Fourier transform results to simulate Shor's algorithm in Chapter 5 under the presence of dephasing noise and quantify the affect on the success of the algorithm. Finally, we simulate Grover's algorithm with matrix product operators in chapter 6. We analyse the complexity of the matrix product operator of the oracle as well as simulating the algorithm for up to 40 qubits with up to 15 X, Y and Z errors individually and with depolarizing noise.

Chapter 2

Quantum Computing

2.1 Quantum bits and gates

The basic unit of information in classical computing is the bit: it is a digital unit that can take the values of 0 or 1 (sometimes called off and on) The analogous unit of information in quantum computing is the quantum bit, or qubit. This is a quantum system in a two dimensional Hilbert space. As such, it can be said to be in a superposition of the 0 and 1 states:

$$|\psi\rangle = a|0\rangle + b|1\rangle, \qquad |a|^2 + |b|^2 = 1. \tag{2.1}$$

Measurement of the qubit in the computational basis returns 0 with probability $p_0 = |a|^2$ and 1 with probability $p_1 = |b|^2$. Similarly, a larger quantum system of d qubits can be considered as a 2^d dimensional qudit, a unit of quantum information with 2^d possible values.

A single quantum bit has several features that are inherently quantum mechanical:

- A quantum bit is in a superposition of basis states, so can be said to be to some degree in both of the quantum states $|0\rangle$ and $|1\rangle$. This can bring about non-determinism in performing a calculation using a quantum bit.

- A quantum bit has a phase between the coefficients of the $|0\rangle$ and $|1\rangle$ basis states, as well as a global phase. The relative phases can be exploited.

It is in scaling up from a single qubit to many qubits that the possible benefits of quantum computing become clear. With n classical bits, there are 2^n possible distinct bit strings. A computer of n bits may be in any of these states. With n qubits, the 2^n distinct bit strings are each a basis state of the overall quantum system. A quantum computer of n qubits can thus exist in any superposition of these states:

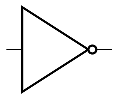
$$|\psi\rangle = \sum_{i_1,\ldots,i_n} c_{i_1,\ldots,i_n}|i_1,\ldots,i_n\rangle, \qquad \sum_{i_1,\ldots,i_n} |c_{i_1,\ldots,i_n}|^2 = 1, \tag{2.2}$$

where i_j is the state of the jth qubit. Having a large superposition of basis states allows a computation to be carried out on each basis state at the same time. This is known as quantum parallelism. However, the probabilistic nature of quantum systems means that there would be no way to efficiently obtain the coefficient of every basis state after performing a computation. A way that information can be extracted from a large quantum system is by interfering the different coefficients c_i. Doing so allows a measurement to extract a piece of information influenced by some or all of the coefficients present. The task of quantum algorithms research is to find ways to exploit features of large quantum superpositions such as these to efficiently perform computational tasks.

Classically, bits are manipulated and interact with Boolean logic gates. In quantum computing, qubits are controlled via unitary quantum gates. In both cases these gates can be drawn in circuit diagrams, with a horizontal line corresponding to one bit (qubit). Gates acting on a given bit (qubit) are placed over the corresponding line. In these diagrams (particularly for quantum circuits), time flows from left to right.

With a single bit, the only classical logic gate is the NOT gate which performs the logical NOT operation $x \rightarrow \sim x$:

The analogous quantum gate is the Pauli X gate. However, a one qubit



Input	Output
A	$\sim A$
1	0
0	1

Figure 2.1: The circuit element and truth table for the classical NOT gate

quantum gate can encode any valid unitary operation on a single qubit. As such, there are an infinite number of one qubit gates. To conceptualise these gates, a one qubit state $|\psi\rangle$ is often parameterised by two angles θ and ϕ:

$$|\psi\rangle = \cos(\theta/2)|0\rangle + \exp i\phi \sin(\theta/2)|1\rangle. \tag{2.3}$$

Note that this parameterisation does not include all of the information needed to uniquely specify a quantum state, but in quantum information we ignore global phases. Using the two angles θ and ϕ as the two angles in a circular coordinate system, the state may be considered as existing on a sphere of radius one called the Bloch sphere, shown in figure 2.2.

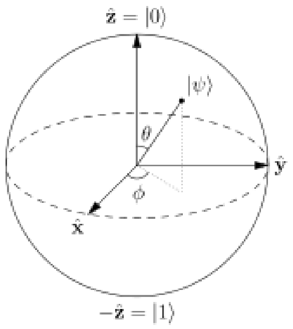


Figure 2.2: The Bloch sphere, source: [80].

Using the parameterisation (2.3), one qubit gates can be considered to be composed of rotations around two of the three perpendicular axes of the Bloch sphere. The Pauli gates X, Y and Z gates represent π rotations around the $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ and $\hat{\mathbf{z}}$ axes respectively, and are shown in figure 2.3. Other

$$-\boxed{X}- \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad -\boxed{Y}- \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix} \qquad -\boxed{Z}- \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

Figure 2.3: The circuit symbols and matrices of the one qubit Pauli gates.

rotations of angle γ around an axis may be formed through exponentiation of the corresponding Pauli matrix:

$$R_x(\gamma) = e^{\frac{-i\gamma X}{2}} X \tag{2.4}$$

$$R_y(\gamma) = e^{\frac{-i\gamma X}{2}} Y \tag{2.5}$$

$$R_z(\gamma) = e^{\frac{-i\gamma X}{2}} Z \tag{2.6}$$

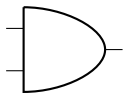
Together with the identity matrix I the Pauli matrices also form a complete basis for one qubit quantum gates.

As well as the Pauli gates, quantum circuits are often written with two additional standard gates: the Hadamard (H) gate and the phase shift (R_θ) gate. These are shown in figure 2.4. We note that the Z gate is a phase shift gate with angle $\theta = \pi$.

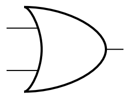
$$-\boxed{H}- \quad \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad -\boxed{R_\theta}- \quad \begin{bmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{bmatrix}$$

Figure 2.4: The circuit symbols and matrices of the Hadamard and phase shift gates.

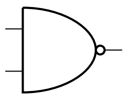
Interactions betdween classical bits are brought about by two bit gates. There are a range of classical logic gates acting on two bits, such as AND, OR, XOR, NOR and NAND. The NAND gate is universal for classical computation. This means any gate with any number of bits can be constructed from only NAND gates, although ancillary bits may be required.



Input		Output
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1



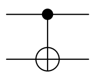
Input		Output
A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



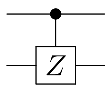
Input		Output
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Figure 2.5: The circuit elements and truth tables for the classical logical AND, OR and NAND gates

Quantum mechanical two qubit gates are, like one qubit gates, unitary operations. An implication of this is that quantum mechanical gates have to be reversible, and so the number of outputs must be equal to the number of inputs. This is an important distinction from classical two bit gates, which often have two inputs but only one output. We show two two-qubit gates in figure 2.6, the controlled not (CNOT) gate and the controlled Z (CZ) gate. The behaviour of these gates, and other controlled gates, can be understood as only applying the gate to the second qubit if the state of the first qubit is one. This classical behaviour description generalises to a quantum mechanical description by applying the gate individually to each basis state and then adding the results to obtain a new quantum state. Another very important two qubit gate is the SWAP gate, which swaps the



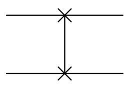
$$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\\0 & 0 & 1 & 0\end{bmatrix}$$



$$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 1 & 0 & 0\\0 & 0 & 1 & 0\\0 & 0 & 0 & -1\end{bmatrix}$$

Figure 2.6: The circuit symbols and matrices for the two qubit quantum gates CNOT and CX.

states of two qubits. This gate is shown in figure 2.7

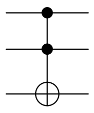


$$\begin{bmatrix}1 & 0 & 0 & 0\\0 & 0 & 1 & 0\\0 & 1 & 0 & 0\\0 & 0 & 0 & 1\end{bmatrix}$$

Figure 2.7: The two qubit SWAP gate

It is possible to describe a variety of gates acting on three qubits and higher, however for simplicity we only refer to controlled gates. In this case, a single qubit gate is applied to one of the qubits if the state of every control qubit is one. The CCNOT gate, in which two qubits are used to control a single X gate, is shown in figure 2.8 and is called a Toffoli gate.

While it is possible to describe an infinite number of gates acting on any number of qubits, it is only necessary to describe a few in order to reproduce any quantum computation. Any classical logic circuit may be produced out of only NOT and AND gates, out of only NAND gates or out of only NOR gates. Note that this statement does not say that it is possible to find an efficient decomposition out of a universal classical gate set, only that a decomposition exists. The three-qubit Deutsch gate is universal



$$\begin{bmatrix} 1&0&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0\\ 0&0&1&0&0&0&0&0\\ 0&0&0&1&0&0&0&0\\ 0&0&0&0&1&0&0&0\\ 0&0&0&0&0&1&0&0\\ 0&0&0&0&0&0&0&1\\ 0&0&0&0&0&0&1&0 \end{bmatrix}$$

Figure 2.8: The circuit symbols and matrices for three qubit Toffoli gate

for quantum compuation [81], and if multiple gates are allowed to form a universal set, then any quantum computation on any number of qubits may be expressed exactly as a product of only one qubit gates and the CNOT gate [82]. Indeed, any computation can be expressed to arbitrary accuracy, but not neccessarily exactly, by only the H, Z, CNOT and $\pi/8$ gates (a $\pi/8$ gate applies a phase of $e^{i\pi/8}$ to the $|1\rangle$ state but no phase to the $|0\rangle$ state) This is the standard universal set of quantum gates, but there are many others. As with classical circuits, these are existence rather than efficiency results. An efficiency result for two qubit unitaries is that using the standard universal set, it is possible to approximate an arbitrary two qubit gate to accuracy ϵ using $O(\log(1/\epsilon))^c$ standard gates, where c is a constant [83]

2.2 Grover's Algorithm

Grover's algorithm is a quantum mechanical algorithm for searching a database. It was first proposed by Grover [5, 6] and extended by Boyer et al [84] to search problems with more than one solution. We will summarise the action of Grover's algorithm following the treatment of [85].

Grover's algorithm deals with a search problem in the following format: the database to be searched consists of the $N = 2^n$ computational basis states in a n qubit system. Given this database, we define a function $f(x)$ which returns 1 if x is one of M solutions to the search problem and 0 is x is not. We are provided with an quantum mechanical oracle (a black box) which we consider as an operator O which applies the following transformation to a state:

$$|x\rangle|q\rangle \xrightarrow{O} |x\rangle|q \oplus f(x)\rangle, \tag{2.7}$$

where $\oplus$ represents addition modulo 2. We consider the oracle to take time $O(1)$ to execute and so ask how many applications of the oracle are required to determine a solution to the search problem. To find a solution out of N elements in an unstructured database classically requires $O(N)$ classical oracle applications, while Grover's algorithm allows the same task to be performed with $O(\sqrt{N})$ quantum mechanical oracle applications. Grover's algorithm thus provides a quadratic speedup over classical methods.

Before we describe the operation of Grover's algorithm, we note a simplification to our notation. If we place a qubit in the state $|-\rangle = 1/\sqrt{2}\,(|0\rangle - |1\rangle)$, then $|- \oplus 0\rangle = |-\rangle$ and $|- \oplus 1\rangle = -|-\rangle$, and so adding q modulo 2 to the state of this qubit for an integer q is equivalent to multiplying the qubit by a phase of $(-1)^q$. By placing the second qubit of (2.7) in the state $|-\rangle$, we can thus consider the oracle as an operator which applied a phase of -1 to all computational basis states in the first qudit which are solutions while leaving unchanged all those basis states which are not:

$$|x\rangle \xrightarrow{O} (-1)^{f(x)}|x\rangle. \tag{2.8}$$

Grover's algorithm consists of initial state preparation and the action of an operator called the Grover iteration m times. To prepare a quantum input state for Grover's algorithm, we take the computational basis state $|0\rangle$ of the overall system, and apply a Hadamard gate to each qubit:

$$|0\rangle \ldots |0\rangle \xrightarrow{H^n} \frac{1}{N}\,(|0\rangle + |1\rangle) \ldots (|0\rangle + |1\rangle) = \frac{1}{N}\sum_{i=1}^{N} |i\rangle. \tag{2.9}$$

The input state to Grover's algorithm, which we refer to as $|\psi\rangle$, is thus an equal superposition of all basis states.

Grover's algorithm requires repeated applications of the Grover iteration. To perform a Grover iteration, we first apply to oracle to the entire quantum state. We then apply an operator called inversion about the mean, which can be written as $I - 2|\psi\rangle\langle\psi|$. As such, we write the Grover iteration as $G = (I_2|\psi\rangle\langle\psi|)\,O$.

To illustrate the affect of Grover's algorithm, we denote the set of solutions to the search problem by $\{s\}$, and write a new basis for the state of the quantum system:

$$|\alpha\rangle = \frac{1}{\sqrt{N-M}} \sum_{x \notin \{s\}} |x\rangle, \quad |\beta\rangle = \frac{1}{\sqrt{N}} \sum_{x \in \{s\}} |x\rangle. \tag{2.10}$$

It is straightforward to write the initial state $|\psi\rangle$ in this basis:

$$|\psi\rangle = \sqrt{\frac{N-M}{N}}|\alpha\rangle + \sqrt{\frac{M}{N}}|\beta\rangle, \tag{2.11}$$

and so the initial state is spanned by the two basis states $\{|\alpha\rangle, |\beta\rangle\}$. If we consider the quantum state to fall on a unit circle in a basis spanned by these two states (α, β), it is clear that the oracle performs a reflection of the state about the α axis. After this, the inversion about the mean operator performs a reflection about the state $|\psi\rangle$. The net result of these two reflections is a rotation in the (α, β) plane towards the β axis (corresponding to a quantum state composed entirely of solutions to the search problem), and away from the α plane.

If we set $\sin(\theta/2) = \sqrt{M/N}$, then $|\psi\rangle = \cos(\theta/2)|\alpha\rangle + \sin(\theta/2)|\beta\rangle$. We can show that applying a Grover iteration amounts to a rotation of the state in the (α, β) plane of θ. As such, applying m Grover iterations will produce the state:

$$G^m|\psi\rangle = \cos\left(\left(m + \frac{1}{2}\right)\theta\right)|\alpha\rangle + \sin\left(\left(m + \frac{1}{2}\right)\theta\right)|\beta\rangle \tag{2.12}$$

Repeated applications of the Grover iteration thus rotate the state close to $|\beta\rangle$. The coefficient of the state in the basis state $|\beta\rangle$ will be maximised by setting $(m + 1/2)\,\theta \approx \pi/2$. It follows that $m = CI\left(\frac{1}{2}\left(\frac{\pi}{\theta} - 1\right)\right)$. By taking the large N/M limit, we may set $\theta \approx 2\sqrt{M/N}$, and so the optimal number of iterations is $m \approx CI\left(\frac{\pi}{4}\sqrt{\frac{N}{M}} - \frac{1}{2}\right)$. We may then find an upper bound for this expression (which in the large N/M limit is a good approximation) $m \approx \lceil \frac{\pi}{4}\sqrt{\frac{N}{M}} \rceil$. After this many Grover iterations, the state will be very close to $|\beta\rangle$ (indeed the coefficient of $|\alpha\rangle$ will decrease quickly as N/M increases) and so be measuring each of the qubits, we have a very high probability of obtaining a solution to the search problem. We thus find that, as stated at the start of this section, the number of oracle calls to obtain a solution to the search problem is $O(\sqrt{N/M})$, which reduces to $O(\sqrt{N})$ if there is only one solution to the search problem, which is classically the most difficult case.

It has been shown that Grover's algorithm is asymptotically the most optimal search algorithm. That is, there are no algorithms that take less oracle calls than $O(\sqrt{N/M})$ [86, 87]. Grover's algorithm has a weakness that the number of solutions is required to be known before the algorithm is run. However, modifications for the algorithm exist which remove this requirement [88, 89].

2.3 Shor's Algorithm

Shor's algorithm is an algorithm to factorise large numbers which are the multiples of two primes. Many modern cryptographic algorithms are founded on the difficulty of this problem, and so being able to run Shor's algorithm in a scalable way would invalidate these algorithms and the security of the data they protect. The time required to perform the fastest known classical algorithms for factorising large numbers scales exponentially with the number of bits in the numbers [90]; however, Shor's algorithm performs this factorisation in an amount of time which is polynomial in the number of bits. As such, Shor's algorithm provides an exponential speedup over the fastest known classical algorithm for the same problem. For this reason, there has been much interest in and study of this algorithm since its conception by Shor [2, 3].

The foundation of Shor's algorithm is the fact that factorising large numbers can be reduced to order finding. That is, for two integers a and N, the order of a modulo N is the smallest positive integer r such that $a^r \mod N = 1$. Order finding is the problem of finding this order for a given set of numbers a, N. The problem of factorising a number N is equivalent to finding a non-trivial solution to the problem $x^2 \mod N = 1$, where $x \neq \pm 1 \mod N$. If we know $a^r \mod N = 1$, r is even and $r \neq \pm 1 \mod N$ then we can take $x = a^{r/2} \mod N$ which will provide a non-trivial solution to $x^2 \mod N = 1$. In this case we can compute $\gcd(x^{r/2} - 1, N)$ and $\gcd(x^{r/2} + 1, N)$, one of which will be a factor for N. Shor's algorithm operates by choosing a random value a in the range $1 \ldots N$ and calculating the $\gcd(a, N)$. The value $\gcd(a, N)$ will either be greater than 1, providing a factor for N, or 1, in which case a and N are co-prime. If a and N are co-prime, the order of a modulo N is calculated. The probability that the order r will be even and will fulfill the above criteria is at least $1 - 1/2^m$, where m is the number of prime factors of N [85]. In this case, a factor will be provided for Shor's algorithm. If this is not the case, the algorithm fails and a new co-prime is chosen.

To factorise a number N with co-prime a by Shor's algorithm, $3n$ qubits are required where $n = \lceil \log_2(N) \rceil$. These qubits are divided into two registers, the upper register which is composed of $2n$ qubits, and the lower register which is composed of the remaining n qubits. We consider the bottom register as a 2^n dimensional qudit which begins in the quantum state $|1\rangle$ and each of the top register's qubits individually. Each of the top register qubits start the computation in the state $|0\rangle$ before having a Hadamard gate applied to them.

Once the initial quantum state has been prepared, Shor's algorithm proceeds in two stages: modular exponentiation and the inverse quantum Fourier transform. To perform modular exponentiation, we require a gate which operates on the lower register as $U|x\rangle = |ax \mod N\rangle$. We then perform a series of quantum gates whose end result is an operator V with the action $V\left(|j\rangle|u\rangle\right) = |j\rangle U^j|u\rangle$. As the quantum computer after initialisation is in the state $2^{-2n}\sum_i |i\rangle|1\rangle$, this state of the computer will be transformed to:

$$|\phi\rangle = 2^{-n}\sum_j |j\rangle\big|a^j \mod N\big\rangle. \tag{2.13}$$

In order to create the operator V, we require the ability to perform a series of gates U^{2^j} for $0 < j < 2n - 1$. Given these gates, we condition the gate U^{2^0} on the first qubit in the upper register, U^{2^1} on the second qubit in the upper register and so on. A circuit performing this operation is shown in figure 2.9. The final state of the system after performing $2n$ controlled U gates will be (2.13).

Once modular exponentiation has been performed, we use the quantum Fourier transform (QFT) to recover r. To show how the QFT operating on the state (2.13) allows this, we follow the introduction to Shor's algorithm in [85]. The eigenvectors of U can be be shown to be:

$$|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left(-\frac{2\pi i s k}{r}\right) \left| x^k \mod N \right\rangle \tag{2.14}$$

In this case it is possible to show that $1/\sqrt{r}\sum_{s=0}^{r-1} e^{2\pi i s k/r} |u_s\rangle = \left| x^k \mod N \right\rangle$, and so we may write (2.13) in terms of $|u_s\rangle$:

$$|\phi\rangle = \frac{2^{-n}}{\sqrt{r}} \sum_{s=0}^{r-1} \left(\sum_{j=0}^{2^{2n}-1} e^{2\pi i s j/r} |j\rangle \right) |u_s\rangle \tag{2.15}$$

The quantum Fourier transform (QFT) is the operator:

$$\text{QFT} = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} e^{2\pi i j k/N} |j\rangle\langle k|. \tag{2.16}$$

We will provide more detail about the QFT and its circuit decomposition in section 2.4. By running the QFT in reverse, we perform the inverse QFT (IQFT), which will perform the transformation $2^{n/2}\sum_{j=0}^{2^n-1} e^{2\pi i j\phi/2^n}|j\rangle \to \left|\widetilde{\phi}\right\rangle$, where $\widetilde{\phi}$ is a n qubit approximation of ϕ. If ϕ can be exactly represented in binary with n bits, then $\widetilde{\phi} = \phi$.

By performing the IQFT on the top register of our state in the form (2.15) we will thus obtain a state which contains a good estimate of r:

$$\text{IQFT}|\phi\rangle = \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \left|\widetilde{s/r}\right\rangle |u_s\rangle. \tag{2.17}$$

By measuring the bottom register we obtain with a high probability a state which is an integer near to $s/r \cdot 2^{2n}$ for some $s < r$. By performing a continued fractions expansion we can then obtain a list of fractions s/r which could obtain such a measurement result. This list of fractions yields a set of r values which can be tested to see if they produce a factor of N. If none of the r values are correct then the algorithm must be re-run; however, there is in general a low probability that the algorithm will fail. Scaling arguments which take into account the probability of success in each individual iteration of Shor's algorithm and optimisations of the algorithm find that the time taken to factorise a number of n bits with Shor's algorithm scales with $O(n^2 \log(n) \log(\log(n)))$ [3].

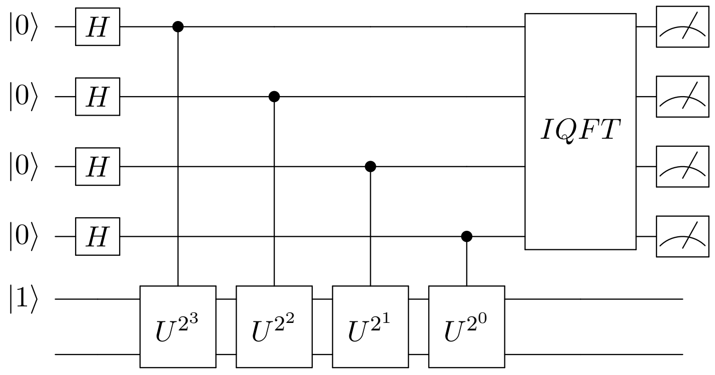


Figure 2.9: A quantum circuit to perform Shor's algorithm for $n = 2$ with six qubits.

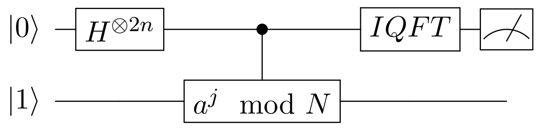


Figure 2.10: The overall action of Shor's algorithm. The top qudit here represents the entire top register and the bottom qudit the bottom register.

Our explication of Shor's algorithm has so far focused on the operation of the algorithm broadly. In this way, we have assumed that U^{2^j} gates are available and that qubits arbitrarily far apart can be interacted. However, in realistic quantum computing architectures there are restrictions on which qubits can interact and only a universal set of gates are available. Specific quantum architectures include those in which qubits are arranged linearly and only nearest neighbours can interact (linear nearest neighbour or LNN architectures) and those in which the qubits are arranged in a $2D$ lattice and nearest neighbours in any direction can interact (2D LNN architectures), as well as others. Work has been performed on how to perform all of Shor's algorithm [91], as well as just the modular exponentiation [92, 87, 93, 94, 95, 96] and just the QFT [97], each in a LNN architecture. Each of these decompositions of Shor's algorithm carries a gate overhead, requires additional ancillary qubits, or both. The different decompositions try to minimise either the gate overhead or the number of ancillas, and so which decomposition is chosen depends upon how costly each of these is in a particular quantum computer. The 2D architectures have also been considered for Shor's algorithm [98, 99], as well as architectures which are similar to those considered but allow interactions between further apart qubits or modify the topologies in which qubits are arranged. The simulation techniques we use in this thesis require a linear arrangement of qubits, and as we wish to perform large scale simulations we will use a circuit with complete U^{2^j} gates operating in a LNN quantum computer.

2.4 Quantum Fourier Transform

The standard quantum Fourier transform (QFT) is an integral part of Shor's algorithm, as well as quantum simulation and other quantum algorithms. As stated in section 2.3, the QFT may be written in terms of its action on a basis state $|j\rangle$:

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi ijk/N} |k\rangle. \tag{2.18}$$

By expanding $|j\rangle$ into its binary representation $|j_1, \ldots, j_n\rangle$, we may rewrite (2.18):

$$\begin{aligned} |j_1, \ldots, j_n\rangle \rightarrow 2^{-n/2} &\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right) \left(|0\rangle + e^{2\pi i 0.j_{n-1}j_n}|1\rangle\right) \\ &\ldots \left(|0\rangle + e^{2\pi i 0.j_1 \ldots j_n}|1\rangle\right), \end{aligned} \tag{2.19}$$

where $0.j_m \ldots j_n = j_m/2 + \ldots + j_m/2^{n-m+1}$. As this representation of the QFT makes explicit the action on each qubit,it is possible to directly convert (2.19) into a quantum circuit. This conversion is performed by noting that the output values at each qubit can be regarded as a series of controlled phase rotations of decreasing size applied to the state $|+\rangle$. To this end, we define a phase rotation gate with an appropriate variable angle:

$$R_k = \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{bmatrix}. \tag{2.20}$$

The quantum circuit for the QFT with four qubits using this gate is shown in figure 2.11. Quantum circuits with higher numbers of qubits generalise from this in a straightforward fashion. This quantum circuit was first described by Coppersmith [100].

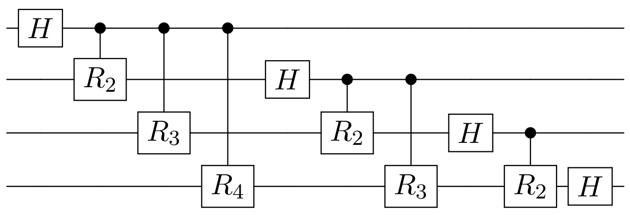


Figure 2.11: The canonical decomposition of the quantum Fourier transform with four qubits. The SWAP gates required to reverse the order of the qubits at the end of the circuit are not shown.

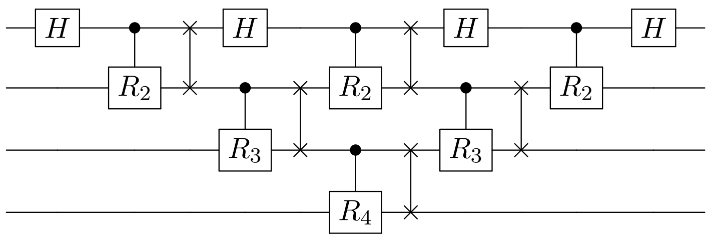


Figure 2.12: A version of the quantum Fourier transform with four qubits using only nearest neighbour quantum gates. The SWAP gates required to reverse the order of the qubits at the end of the circuit are not shown.

As we noted in section 2.3, the QFT can be decomposed into two qubit gates on a linear nearest neighbour architecture. There is more than one approach to how to perform this mapping, but we will use the decomposition of [91] in this thesis. The LNN quantum circuit for the four qubit QFT in this decomposition is shown in figure 2.12.

A number of improvements or modifications of the standard QFT are possible in order to make it easier to perform on a quantum computer. The size of the smallest controlled phase rotation in the QFT decreases exponentially with the number of qubits being transformed. Very small phase rotations would be expected to have little effect on the final output of a quantum circuit but will cause a large increase in the number of universal gates required to implement a circuit. As such, all phase rotations smaller than a certain value are often removed from the QFT. In this case the QFT is referred to as the approximate QFT or AQFT, and the maximum number of phase rotations applying to an individual qubits referred to as the bandwidth. A circuit for the AQFT with five qubits and a bandwidth of two is shown in figure 2.13.

Another modification of the QFT is to replace it with a much simpler

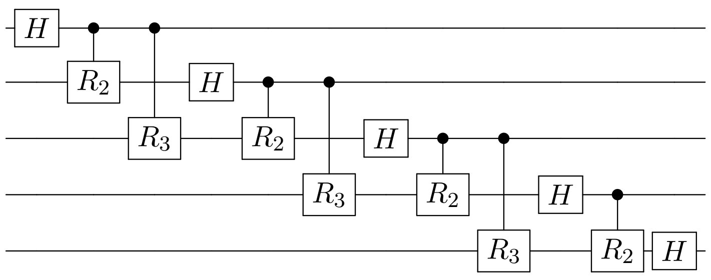


Figure 2.13: The AQFT for five qubits with a bandwidth of two.

circuit called the semiclassical quantum Fourier transform, suggested by Griffiths and Niu [101]. This modification may be performed if each qubit will be measured immediately after the QFT, and so the measurement probabilities of each qubit matter, but the relative phases do not. In this case, we measure each qubit immediately after its final Hadamard gate and then classically condition each controlled phase rotation on the measurement result. A quantum circuit implementing this procedure is shown in figure 2.14 for four qubits. Classically conditioning rather than quantum mechanically conditioning the phase rotation gates significantly reduces the number of coherent quantum gates that must be applied to obtain the correct measurement outcome probabilities.

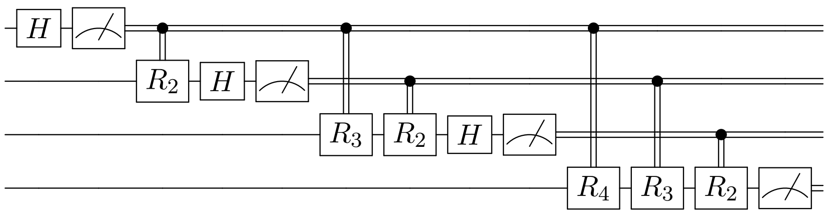


Figure 2.14: The semiclassical QFT with four qubits. Double lines here represent classical information.

2.5 Entanglement

Entanglement is perhaps the feature of quantum mechanics which is the most nonclassical. Two quantum mechanical systems are said to be entanglement if their combined state is not separable. That is, they are entangled if the state cannot be written in the form $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$, where $|\psi_1\rangle$ and $|\psi_2\rangle$ are the states of the two subsystems. If the systems are entangled, then measuring with one system will instantaneously change the quantum mechanical state of the other system. This idea is the source of the Einstein-Podolsky-Rosen (EPR) paradox, and the difference between classical correlations and quantum mechanical entanglement is quantified in Bell's inequalities.

In searching for the source of the speedup of quantum algorithms over their fastest known classical counterparts, we would expect to find nonclassical features of quantum mechanics displayed by the quantum algorithms. As the most nonclassical feature of quantum mechanics, entanglement would seem to be a prime candidate for being the source of, or at least a contributing factor to, the quantum speed-up. This contention is aided by the importance of entanglement to other quantum information applications such as quantum key distribution [102], quantum teleportation [103] and quantum dense coding [104]. Despite this, the question of the usefulness and necessity or otherwise of using highly entangled states for quantum computing remains open. Several results have been published indicating that without entangelement quantum computers can be efficiently simulated [38, 39, 40, 41, 42], but these results rely on pure state quantum computation. It has been shown that algorithms and schemes capable of exponential speed-up exhibit potentially large amounts of entanglement [43, 44, 45].

Discord [51, 52] is another measure of nonclassical correlations between two different subsystems of a quantum system. A class of quantum algorithms have been proposed which use a single qubit with non-zero purity and other qubits in a completely mixed state [32]. These algorithms have an exponential advantage over the best known classical algorithms for certain tasks. They display very small small amounts of entanglement, but large amounts of overall correlation and of discord [48, 49, 53]. Similarly, quantum mechanical computations in mixed states with little or fixed entanglement but a speed-up over classical computations has been found [50] Such results have even been found in pure states [46]. The difficulties inherent in solving the problem of the source of quantum speed-up are increased by the fact that entanglement and discord are difficult to calculate for large mixed states [105].

In light of the debate surrounding the importance of the quantum entanglement, it is clear that entanglement is an important quantity to calculate when simulating algorithms. Calculating the entanglement can also provide insight into how an algorithm functions and how it is affected by noise. In this thesis we will calculate the entanglement in pure state simulations of Shor's algorithm, and so we summarise here a few measures of entanglement. A measure of entanglement is a function which ranks states from the least entangled to the most entangled. In order to be a entanglement measure, a candidate function f must fulfill a set of criteria:

- Separable states have no entanglement, and so an entanglement measure must return 0 on these states. These are pure states of the form $|\psi_1\rangle\otimes|\psi_2\rangle\otimes\ldots\otimes|\psi_n\rangle$ or mixed states of the form $\sum_i \rho_{i_1}\otimes\rho_{i_2}\otimes\ldots\otimes\rho_{i_n}$.
- Entanglement cannot increase under local operations and cloassical communication for any valid local quantum mechanical local operation.
- Entanglement does not change under local unitary operations.

In line with the first criterion above, any two entanglement measures must agree which states are not entangled; however, they may rank entangled states in different orders. In general then, whether one state is more entangled than another depends specifically on which entanglement measure is used. We will now introduce some entanglement measures.

Entropy of entanglement

The first entanglement measure we introduce is the entropy of entanglement [106]. Given a pure state $|\psi\rangle$ divided into two halves a and b through a suitable bipartition, the entropy of entanglement is the von Neumann entropy of the reduced state in each subsystem. That is, $S = -\mathrm{tr}(\rho_a \log(\rho_a)) = -\mathrm{tr}(\rho_b \log(\rho_b))$ where ρ_a and ρ_b are the reduced density matrices of the state $|\psi\rangle$ in the subsystems a and b respectively. If we write one of ρ_a or ρ_b in terms of its eigenvalues $\rho_a = \sum_j \lambda_j |j\rangle\langle j|$ (or similarly for ρ_b), then $S = -\sum_j \lambda_j \log \lambda_b$. To calculate the entropy, we must either be able to take the logarithm of a reduced density matrix or write it in terms of its eigenvalues. For an arbitrary large quantum state, these calculations will be difficult to perform. Many mixed state entanglement measures reduce to the entropy of entanglement when considering pure states. Such measures are called entanglement monotones.

Negativity

The negativity [107] is an entanglement monotone that does not require variational maximisation. It is derived from the PPT criterion, that the partial transpose of an unentangled quantum system must preserve its positivity. The negativity is given by:

$$\mathcal{N}(\rho) = \frac{\|\rho^{T_A}\|_1 - 1}{2}, \tag{2.21}$$

where ρ^{T_A} is the density matrix ρ with subsystem A transposed, and $\|\rho\| = \mathrm{tr}\left(\sqrt{\rho^\dagger \rho}\right)$ is the trace norm of ρ. Equation (2.21) is equivalent to an alternative expression:

$$\mathcal{N}(\rho) = \sum_j \theta_j = \sum_j \frac{|\lambda_j| - \lambda_j}{2}, \tag{2.22}$$

where λ_j are the eigenvalues of ρ, and $\{\theta_j\}$ is the sum of the negative eigenvalues of ρ.

Logarithmic-negativity

Another entanglement measure related to the negativity is the logarithmic-negativity (log-negativity), which is the only entanglement measure we will compute in this thesis. The log-negativity is given by:

$$E_N(\rho) = \log_2\left(\|\rho^{T_A}\|_1\right) = \log_2\left(2\mathcal{N} + 1\right). \tag{2.23}$$

As it is derived from the negativity, the log-negativity is relatively straightforward to compute. It is not convex and so was intially thought to be able to be increased by LOCC (local operations and classical communication) [107]. However, it was later shown to be an entanglement monotone [108]. It provides an upper bound to a more difficult entanglement measure to compute, the distillable entanglement. The distillable entanglement is the number of states which are arbitrarily close to bell states which can be prepared from a state ρ by LOCC, in the asymptotic limit that there are an infinite number of states ρ. The logarithmic negativity is also additive on tensor products (that is $E_N(\sigma \otimes \rho) = E_N(\sigma) + E_N(\rho)$), but may be zero even if a state is entangled.

Geometric measure of entanglement

A geometric measure of entanglement [109, 110] measures the distance of a given state to the nearest unentangled state. More formally:

$$E(|\psi\rangle) = \min_{|\phi_1\rangle,\ldots,|\phi_n\rangle} D\left(|\psi\rangle, |\phi_1\rangle \otimes \ldots \otimes |\phi_n\rangle\right), \tag{2.24}$$

where n is the number of subsystems of $|\psi\rangle$ and D is a measure of distance between states (for example the inner product) which ensures that E has the properties required of an entanglement measure. Unlike the negativity or entropy of entanglement, a geometric measure of entanglement does not require the choice of a particular bipartition of the quantum system, and so is able to calculate multipartite entanglement.

A geometric measure of entanglement can be extended to mixed states with a convex roof construction. That is:

$$E(\rho) = \min_{\mathcal{E}} \sum_i p_i E(|\psi_i\rangle), \tag{2.25}$$

where the minimisation is taken over all sets of pure states $\mathcal{E} = \{p_i, |\psi_i\rangle\}$ which produce $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$. Such a calculation requires variational minimisation over the set of separable states to calculate the entanglement of each $|\psi_i\rangle$, as well as over the set of permissible decompositions of ρ. As such, it is in general quite difficult to calculate.

2.6 Decoherence

The descriptions of the operation of each quantum algorithm rely upon the assumption that individual quantum systems can be addressed, controlled and interacted perfectly. However, physical quantum devices exist in noisy external environments and are very difficult to individually control. As a result, a variety of different kinds of noise and errors can compromise the success of quantum algorithms. These can include interactions with the environment leading to decoherence, imperfect gate application (which may lead to coherent errors), errors in qubit initialization or measurement, qubit loss and qubit leakage. We will first consider the affect of interactions between a quantum computer and its environment.

In a specific quantum system, the influence on the environment can be incorporated into the evolution of the system by starting with an initially uncoupled system and environment $\rho_0 = \rho_s \otimes \rho_e$, evolving them with an entangling operator which reflects their interaction to obtain $\rho_1 = U^\dagger \rho_s \otimes \rho_e U$, and then tracing out the environment to obtain the reduced density matrix of the system $\rho_2 = \mathrm{tr}_e \left(U^\dagger \rho_s \otimes \rho_e U \right)$. If the system interacts in a Markovian fashion with the environment then a similar operation can be carried out at each time step in a quantum evolution and a final state determined after a quantum algorithm, but non-Markovian noise is more difficult to model. In either case, the end result will be that the quantum state after the evolution will be less coherent and the success of the quantum algorithm will be compromised.

To aid the computation of state evolution, the picture we have introduced of taking a separable system and environment, jointly evolving them and then tracing out the environment can be rewritten in the quantum operator formalism. The evolution of a density operator $\rho \in \mathbb{C}^{n \times n}$ can be expressed with a quantum operator $\mathcal{E}$, which is a map from $\mathbb{C}^{n \times n}$ to $\mathbb{C}^{m \times m}$, where m and n may be different. In order to make this an allowable quantum operation, the map must be normalised in the sense that $\mathrm{tr}\left(\mathcal{E}(\rho)\right) \leq 1$, the map must apply linearly, and it must be completely positive (CP). That is, as density matrices must be positive, the outcome of a quantum operation $(\mathcal{I} \otimes \mathcal{E})(\rho_2 \otimes \rho)$ must be positive, where ρ_2 is any state in an additional quantum system.

Choi's theorem says that a completely positive map, such as the quantum operators we have described, may be written in the operator sum notation:

$$\mathcal{E}(\rho) = \sum_k E_k \rho E_k^\dagger, \tag{2.26}$$

where $\{E_k\}$ are a set of operators called the Kraus operators where $\sum_k E_k^\dagger E_k \leq I$. The evolution in (2.26) can be considered as taking ρ and replacing it with $E_k \rho E_k^\dagger / \mathrm{tr}\left(E_k \rho E_k^\dagger\right)$ with probability $\mathrm{tr}\left(E_k \rho E_k^\dagger\right)$ for each k. We can thus think of noisy quantum evolution in the operator-sum representation as a noisy quantum channel in which each of the E_k has a set chance of being applied.

We now describe the approach normally taken to analyse the noisy evolution of circuits. At the simplest level, general one qubit noise-models acting on each qubit are often considered instead of specific noise models motivated by the behaviour of specific physical systems. Correlated noise acting on larger numbers of qubits is physically more unlikely and so is not usually considered. We will consider several one-qubit noisy quantum channels.

Bit flip channel

The bit flip channel has Kraus operators:

$$E_0 = \sqrt{p}I, \quad E_1 = \sqrt{1-p}X, \tag{2.27}$$

and represents an evolution in which with probability p a quantum state will have its bit flipped while with probability $1-p$ it will be unchanged.

Phase flip channel

The phase flip channel has Kraus operators:

$$E_0 = \sqrt{p}I, \quad E_1 = \sqrt{1-p}Z, \tag{2.28}$$

and so has a probability p of the phase of a qubit being flipped.

Bit-phase flip channel

The bit-phase flip channel has Kraus operators:

$$E_0 = \sqrt{p}I, \quad E_1 = \sqrt{1-p}Y = \sqrt{1-p}\,iXZ, \tag{2.29}$$

and encodes a probability of both a bit and phase flip occurring on a qubit with probability p.

Depolarising channel

The depolarising channel is given by a quantum operation $\mathcal{E}$, whose action is given by:

$$\mathcal{E}(\rho) = p\frac{I}{2} + (1-p)\rho. \tag{2.30}$$

The channel thus represents the qubit being replaced by a completely mixed state with a probability p and untouched with probability $1-p$. We can rewrite (2.30) as:

$$\mathcal{E}(\rho) = \left(1 - \frac{3}{4}p\right)\rho + \frac{p}{4}\left(X\rho X + Y\rho Y + Z\rho Z\right), \tag{2.31}$$

and so we can interpret the depolarising channel as having a fixed probability of applying a X, Y or Z matrix to a qubit.

While the first three quantum channels we have described may not seem to adequately encode any likely physical behaviour, they are useful for the analysis of any other single qubit quantum noise. To see this, we note that $\{I, X, Y, Z\}$ constitutes a complete basis for one qubit operators and so any one qubit operator may be expressed as a linear combination of these operators. Some appreciation for the affect of general one qubit noise can thus be gained for the knowledge of how the bit flip, phase flip and bit-phase flip channels affect a quantum computer. In this thesis, we will consider quantum noise which acts independently on each qubit. We will consider the bit flip, phase flip, bit phase flip and depolarising channels. To model the effect of this quantum noise, we will use a technique called error expansion [111, 47]. That is, the evolution of a noisy quantum circuit with independent noise on each qubit may be represented by a sum of the pure state evolutions of quantum circuits with error operators placed at fixed points in the circuits, weighted by their probabilities of occurring. As such, we will perform stochastic pure state evolutions of quantum circuits with fixed probabilities of errors (X, Y or Z gates) occurring at each time step. By summing the probabilities of success of these stochastic pure states for a sufficiently large random sample, we will obtain a good approximation to the results of the mixed state evolution. We note that while such an average gives meaningful information about the behaviour of a noisy circuit, any individual stochastic pure state simulation does not reflect a quantum state reached by the quantum computer.

Other kinds of quantum errors may be modelled by similar processes. For example, the imperfect operation of gates may be coherent, in which case the imperfect gate must be changed but the state will remain pure. Incoherent imperfect gate applications may be simulated stochastically with similar techniques to those used to simulate single qubit noise. That is, an imperfect gate is randomly chosen for each simulation with probability given the the probability the gate has of occurring (this can be obtained from considering the gate as a quantum operation and calculating the Kraus operators). Summing over a large number of these stochastic simulations will allow results about the quantum evolution to be calculated. We will not consider imperfect gates or other forms of error in this thesis, however they may be found in many past results.

While it is inevitable in any physical quantum computer that noise will occur and the computer will not behave perfectly, it is theoretically possible to correct quantum errors and run a quantum computer fault tolerantly. The completeness of the $\{I, X, Y, Z\}$ basis is particularly important for quantum error correction. That is, as each Kraus operator of any noisy channel may be written as a sum of I, X, Z and $Y = iXZ$, it is possible to show that if a quantum computer is robust against X and Z errors, it is robust against any single qubit quantum noise. Error correction codes seek to bring about this robustness and involve encoding logical qubits in larger numbers of physical qubits. We will not simulate error correction codes in this thesis, but they are often simulated in order to determine what probability of qubit failure they are robust against. An introduction to and survey of this field may be found in [56].

Chapter 3

Matrix product states and operators

In this chapter we will provide an introduction to matrix product states and matrix product operators. As these representations are used in the remainder of this thesis to simulate quantum algorithms, we will review how the representations work and how to perform calculations with them. Additionally, we will give a brief introduction to projected entangled pair states, however we these will not be used in later chapters. Matrix product states and operators and projected entangled pair states are examples of tensor networks used to represent quantum systems. A more exhaustive introduction to these networks and survey of the literature regarding them can be found in [112]. Additionally, while they are not relevant to this thesis, many other kinds of tensor networks have been developed to describe and simulate quantum systems. Some examples are correlator product states [113], tree tensor networks [114] and the multiscale entanglement renormalisation ansatz [115, 116, 117].

3.1 Matrix product states

A pure quantum state of n qubits is written as:

$$|\psi\rangle = \sum_{i_1,\ldots,i_n=0}^{1} c_{i_1,\ldots,i_n} |i_1\rangle \otimes \ldots \otimes |i_n\rangle \tag{3.1}$$

We can consider the coefficients $c_{i_1,\ldots,i_n}$ as a vector with 2^n entries. Alternatively, we can also consider c as a tensor with n indices, each corresponding to one qubit and of dimension 2. To recover a coefficient, we set each index to the required value (0 or 1) and the resulting value in the tensor is the coefficient of the quantum state in that computational basis state. The total number of coefficients, as well as the appropriate ordering in the vector, can be derived from this tensor because of its tensor product structure.

A tensor with n indices, each corresponding to one qubit, will contain more information than is needed to uniquely specify a quantum state which is not fully entangled. To illustrate this, consider n qubits, each in a state $a|0\rangle + b|1\rangle$, where both a and b are nonzero:

$$|\psi\rangle = (a_1|0\rangle + b_1|1\rangle) \otimes (a_2|0\rangle + b_2|1\rangle) \otimes \ldots \otimes (a_n|0\rangle + b_n|1\rangle) \tag{3.2}$$

This state is separable, and the state of each of the constituent systems does not depend on the state of any other constituent system. As such, we could represent each qubit individually by a dimension 2 vector, and the overall quantum state would be uniquely specified by n such vectors. This gives a total storage cost of $O(2n)$. However, each of the 2^n computational basis coefficients of this n qubit system will be nonzero, and so a naive representation of the tensor $c_{i_1,\ldots,i_n}$ will require space $O(2^n)$. The full state tensor is clearly a non-optimal way to represent or perform calculations on this state.

In this trivial example (3.2) is the most efficient representation of the state. This is because the state is separable and so has no entanglement. However, a general quantum state will have entanglement and will not be as straightforward to represent. Nonetheless, we can extend the technique used above to a more general way of representing states efficiently. To do this, we note that (3.2) can be written as

$$|\psi\rangle = \sum_{i_1,\ldots,i_n} C^{[1]}_{i_1} C^{[2]}_{i_2} \ldots C^{[n]}_{i_n} |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_n\rangle, \tag{3.3}$$

and so it is clear that the overall tensor $c_{i_1,\ldots,i_n}$ is formed from a product of n tensors (tensors with one dimension are more commonly called vectors). This is a simple example of a matrix product state. We generalise this example by allowing each constituent tensor to have three dimensions, and so we have a product of higher dimensional tensors:

$$|\psi\rangle = \sum_{i_1,\ldots,i_n} \sum_{\alpha_1,\ldots,\alpha_{n-1}} \Gamma^{[1]\alpha_1}_{i_1} \Gamma^{[2]\alpha_1\alpha_2}_{i_2} \ldots \Gamma^{[n-1]\alpha_{n-2}\alpha_{n-1}}_{i_{n-1}} \Gamma^{[n-2]\alpha_{n-1}}_{i_n} \\ |i_1\rangle \otimes |i_2\rangle \otimes \ldots \otimes |i_n\rangle \tag{3.4}$$

Each of the α_i here is an ancillary index that is summed over on two adjacent tensors. This operation is simply tensor multiplication and is more familiar in dimension 2 tensors as matrix multiplication. Together, the ancillary indices specify a one-dimensional ordering of the constituent systems. As such, this way of writing a quantum state is most appropriate for writing states in which such an ordering is natural. It is possible to phrase quantum computing entirely in terms of nearest neighbour interactions, and so in this case a one dimensional ordering of the quantum states is reasonable. The dimensionality of a quantum computing system is important for other properties such as the number of qubits required or the ability to implement error correction, so higher dimensional orderings will be required in these cases.

Within a one dimensional ordering, each ancillary index (henceforth referred to as a bond) specifies a bipartition of the quantum state. Bond sizes larger than one can be used to contain information about entanglement within the state.

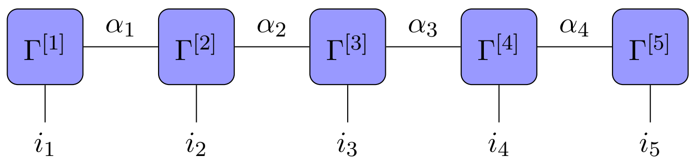


Figure 3.1: A pictorial representation of a MPS with five qubits.

Matrix product states and other tensor networks are often represented pictorially. Such a diagram is shown in figure 3.1. In this diagram, a tensor is represented by a square (or other filled geometric shape) with a number of lines emanating from it. Each of these lines represent one index in the tensor. Lines linking two separate tensors correspond to indices that are summed over to obtain a final coefficient. Lines which adjoin only one tensor are indices which are not summed over. In this case these indices represent the quantum states of the individual qubits.

3.1.1 Freedom in the MPS representation

In the general statement of a matrix product state (3.4) we have not specified the sizes of the bonds being summed over. It is clear that given this omission, there are a large number of different MPSs conforming to (3.4) which will encode the same quantum state. Indeed, given an MPS written as

$$|\psi\rangle = \sum_{i_1,\dots,i_n} A^{[1]}_{i_1} A^{[2]}_{i_2} \dots A^{[n-1]}_{i_{n-1}} A^{[n]}_{i_n} |i_1, i_2, \dots, i_{n-1} i_n\rangle, \tag{3.5}$$

we can change the tensors while leaving the state invariant by multiplications:

$$\begin{aligned} |\psi\rangle = \sum_{i_1,\dots,i_n} & A^{[1]}_{i_1} X^{[1]} \left(X^{[1]}\right)^{-1} A^{[2]}_{i_2} X^{[2]} \left(X^{[2]}\right)^{-1} \dots \\ & A^{[n-1]}_{i_{n-1}} X^{[n-1]} \left(X^{[n-1]}\right)^{-1} A^{[n]}_{i_n} |i_1, i_2, \dots, i_{n-1} i_n\rangle. \qquad (3.6) \\ = \sum_{i_1,\dots,i_n} & B^{[1]}_{i_1} B^{[2]}_{i_2} \dots B^{[n-1]}_{i_{n-1}} B^{[n]}_{i_n} |i_1, i_2, \dots, i_{n-1} i_n\rangle, \qquad (3.7) \end{aligned}$$

where, for example, $B^{[2]} = X^{[1]\,-1} A^{[2]}_{i_2} X^{[2]}$. The matrices $X^{[i]}$ do not need to be square, but they do need to be right invertible. That is:

$$X^{[i]}\, X^{[i]\,-1} = \mathbb{I},\, X^{[i]\,-1}\, X^{[i]} \neq \mathbb{I} \tag{3.8}$$

Allowing these matrices to be non-square allows the sizes of the bonds to change. Using a non-square matrix will increase bond sizes and will lead to less optimal encodings of quantum states. This freedom was first described in [118], where it is also shown that multiplying by matrices as shown here is a completely general description of the freedom present in the MPS representation.

Schmidt Decomposition

The optimal bond size in a MPS is motivated by considering the Schmidt decomposition of a state. Given a state $|\psi\rangle$ and a bipartition of the its constituent qubits $A : B$ which is associated with two Hilbert spaces H_A

and H_B, the state can be written in the form:

$$|\psi\rangle = \sum_i \lambda_i |\alpha_i\rangle \otimes |\beta_i\rangle, \tag{3.9}$$

where $\{|\alpha_i\rangle\}$ and $\{|\beta_i\rangle\}$ are orthonormal bases of vectors in H_A and H_B respectively. The coefficients λ_i are non-negative and are known as the Schmidt coefficients. The number of nonzero coefficients is known as the Schmidt rank of the state. The vectors $\{|\alpha_i\rangle\}$ and $\{|\beta_i\rangle\}$ are known as the Schmidt vectors. The orthogonality of the basis vectors in (3.9) places an upper bound on the Schmidt rank of any state of $\max(d_A, d_B)$, where d_A and d_B are the dimensions of H_A and H_B respectively.

The Schmidt rank of a pure state $|\psi\rangle$ is a measure of the entanglement of the state. A state with Schmidt rank one is separable and so has no entanglement while a state with higher Schmidt rank requires additional parameters to be specified in order to encode the state. Critically, the Schmidt rank of a state $|\psi\rangle$ is the *minimum* number of coefficients needed to write the state in the form (3.9). A MPS encodes a state in a one dimensional form and so every bond corresponds to a bipartition of the state. As such, the Schmidt rank provides the minimum bond size required to represent a state as a MPS at each bipartition. The Schmidt decomposition thus provides a way of encoding a MPS maximally efficiently. In order to outline an algorithm to write a MPS in this way for a given state, we will first review the relationship between the Schmidt rank and the singular value decomposition (SVD).

The SVD is used in order to prove that every state can be written in the form (3.9). To review this relationship, we first write a state $|\psi\rangle$ over a bipartition $A : B$ as a two-dimensional tensor:

$$|\psi\rangle = \sum_{i,j} C_{i,j} |a_i\rangle \otimes |b_j\rangle, \tag{3.10}$$

where $\{|a_i\rangle\}$ and $\{|b_j\rangle\}$ form orthonormal bases for H_A and H_B respectively. By using the SVD, we can express the $d_A \times d_B$ complex matrix $\mathbf{C}$ as the product of the form:

$$\mathbf{C} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^*, \tag{3.11}$$

where $\mathbf{U}$ is a $d_A \times d_A$ complex unitary matrix, $\mathbf{\Sigma}$ is a $d_A \times d_B$ real non-negative diagonal matrix and $\mathbf{V}^*$ is a $d_B \times d_B$ complex unitary matrix. The diagonal entries of $\mathbf{\Sigma}$ are known as the singular values of the matrix $\mathbf{C}$. Because of the size of C, there are $\min(d_A, d_B)$ non-negative singular values. We can thus rewrite (3.11) as

$$C_{j,k} = \sum_i U_{ji} \Sigma_i V^*_{ik}. \tag{3.12}$$

In this form, we can draw a connection between (3.9) and (3.11):

$$\lambda_i = \Sigma_i, \qquad |\alpha_i\rangle = \sum_j U_{ju} |a_j\rangle, \qquad |\beta_i\rangle = \sum_k V^*_{ik} |b_k\rangle. \tag{3.13}$$

And so the Schmidt decomposition is a rephrasing of the singular value decomposition. That $\{|\alpha_i\rangle\}$ and $\{|\beta_i\rangle\}$ are orthonormal sets follows simply from the properties of the matrices $\mathbf{U}$ and $\mathbf{V}^*$.

Canonical Representation

The canonical representation of a MPS and its use in simulating quantum circuit were first proposed by Vidal in [40]. We follow his approach here. We start with a state which we wish to write as a MPS:

$$|\psi\rangle = \left|\Psi^{[1\dots n]}\right\rangle = \sum_{i_1,\dots,i_n=0}^{1} c_{i_1,\dots,c_n} |i_0\rangle \otimes \dots \otimes |i_n\rangle. \tag{3.14}$$

We first perform a SVD between the first qubit and the other qubits:

$$|\psi\rangle = \sum_{\alpha_1} \lambda^{[1]}_{\alpha_1} \left|\Psi^{[1]}_{\alpha_1}\right\rangle \left|\Psi^{[2\dots n]}_{\alpha_1}\right\rangle. \tag{3.15}$$

Writing the Schmidt vectors of the first qubit $\left|\Psi^{[1]}_{\alpha_1}\right\rangle$ in the computational basis of the first qubit (which occurs naturally if the SVD is performed in this basis) gives:

$$|\psi\rangle = \sum_{\alpha_1} \Gamma^{[1]\alpha_1}_{i_1} \lambda_{\alpha_1} |i_1\rangle \left|\Psi^{[2\dots n]}_{\alpha_1}\right\rangle. \tag{3.16}$$

The following three steps are then iterated to obtain a MPS:

1. Each Schmidt vector $\left|\Psi^{[2\ldots n]}_{\alpha_1}\right\rangle$ is written in the local basis of its left-most qubit:

$$\left|\Psi^{[2\ldots n]}_{\alpha_1}\right\rangle = \sum_{i_2} |i_2\rangle \left|\tau^{[3\ldots n]}_{\alpha_1 i_2}\right\rangle. \tag{3.17}$$

2. We perform a Schmidt decomposition on each of the resulting vectors $\left|\tau^{[3\ldots n]}_{\alpha_1 i_2}\right\rangle$. We thus write each τ vector in terms of the largest number of nonzero singular values resulting from these decompositions:

$$\left|\tau^{[3\ldots n]}_{\alpha_1 i_2}\right\rangle = \sum_{\alpha_2} \Gamma^{[2]i_2}_{\alpha_1\alpha_2} \lambda^{[2]}_{\alpha_2} \left|\Psi^{[3\ldots n]}_{\alpha_2}\right\rangle. \tag{3.18}$$

3. Substitute (3.17) and (3.18) into the previous expression (3.16):

$$|\psi\rangle = \sum_{\alpha_1}\sum_{\alpha_2} \Gamma^{[1]\alpha_1}_{i_1} \lambda_{\alpha_1} \Gamma^{[2]\alpha_1\alpha_2}_{i_2} \lambda_{\alpha_2} |i_1\rangle |i_2\rangle \left|\Psi^{[3\ldots n]}_{\alpha_2}\right\rangle. \tag{3.19}$$

If we iterate these steps, we can remove qubits from the larger right hand side tensor one by one until we are left with a MPS representation:

$$\begin{aligned} |\psi\rangle = \sum_{i_1,\ldots,i_n} \sum_{\alpha_1,\ldots,\alpha_{n-1}} & \Gamma^{[1]\alpha_1}_{i_1} \lambda_{\alpha_1} \Gamma^{[2]\alpha_1\alpha_2}_{i_2} \lambda_{\alpha_2} \ldots \\ & \Gamma^{[n-1]\alpha_{n-2}\alpha_{n-1}}_{i_{n-1}} \lambda_{\alpha_{n-1}} \Gamma^{[n]\alpha_{n-1}}_{i_n} |i_1,\ldots,i_n\rangle, \end{aligned} \tag{3.20}$$

and so any state $|\psi\rangle$ of maximal Schmidt rank χ can be written as a MPS with bond rank at most χ.

The form (3.20) is slightly different from the ones we have quoted earlier in that the bonds now contain the vectors of their singular values. However, the forms can easily be made equivalent by multiplying the vectors into the tensors either to their left or their right. Additionally, it is advantageous to leave the vectors in the bonds because the singular values along each bipartition provide information about the state at that bipartition. In particular, the size of each of the singular values shows how much entanglement is contained in that particular bond element. As such, the canonical representation provides us an easy way of making our MPS smaller on a given bipartition while keeping the most pertinent information to the dynamics of the system. In this representation, we delete every bond element which is smaller than a certain size. We must then renormalise the bond as follows:

$$\sum_{\alpha_i=1}^{d} \Gamma_i^{[i]\alpha_{i-1}\alpha_i} \lambda_{\alpha_i}^{[i]} \Gamma_{i+1}^{[i+1]\alpha_i\alpha_{i+1}} \to \sum_{\alpha_i=1}^{d'} \Gamma_i^{[i]\alpha_{i-1}\alpha_i} \lambda_{\alpha_i}^{\prime[i]} \Gamma_{i+1}^{[i+1]\alpha_i\alpha_{i+1}}, \tag{3.21}$$

where $\lambda_j^{\prime[i]} = \lambda_j^{[i]} \left(\sum_{\alpha_i=1}^{d} \lambda_{\alpha_i}^{[i]2}\right) / \left(\sum_{\alpha_i=1}^{d'} \lambda_{\alpha_i}^{[i]2}\right)$.

We have made no assumptions about the quantum state in describing the algorithm to decomposed it as a MPS, and so any quantum state can be written in the form (3.20). The MPS form obtained after this decomposition is often described as the canonical form for open boundary condition MPSs (OBC-MPS). In terms of the freedoms available in the MPS representation, it is shown in [40, 118] that if a state is written as a MPS with the format (3.5), then if the following gauge conditions are fulfilled the state is in canonical form:

1. $\sum_i A_i^{[m]} A_i^{[m]\dagger} = \mathbb{I}_{d_m}$ for all $1 \leq m \leq n$.
2. $\sum_i A_i^{[m]\dagger} \Lambda^{[m-1]} A^{[m]_i} = \Lambda^{[m]}$, for all $1 \leq m \leq n$.
3. $\Lambda^{[0]} = \Lambda^{[n]} = 1$ and $\Lambda^{[m]}$ is a $d_{m+1} \times d_{m+1}$ diagonal matrix of full rank and with trace one.

It is also shown in [118] that any OBC-MPS in the form (3.5) can be multiplied by matrices as detailed above in section 3.1.1 to obtain a OBC-MPS in canonical form.

3.1.2 Calculations with MPSs

Contraction

Given a MPS, any required coefficient in the computational basis may be found by setting each of the open indices to specify which coefficient is required, and then contracting the tensors one by one. If we take the general MPS form specified in (3.4) and require the coefficient of the computational basis state $|j_1 j_2 \ldots j_n\rangle$, this can be written as:

$$c_{j_1,j_2,\ldots,j_n} = \sum_{\alpha_1,\ldots,\alpha_n} \Gamma_{j_1}^{[1]\alpha_1} \Gamma_{j_2}^{[2]\alpha_1\alpha_2} \Gamma_{j_3}^{[3]\alpha_2\alpha_3} \ldots \Gamma_{j_n}^{[n]\alpha_{n-1}} \tag{3.22}$$

$$= \sum_{\alpha_2,\ldots,\alpha_n} \left(\sum_{\alpha_1} \Gamma_{j_1}^{[1]\alpha_1} \Gamma_{j_2}^{[2]\alpha_1\alpha_2} \right) \Gamma_{j_3}^{[3]\alpha_2\alpha_3} \ldots \Gamma_{j_n}^{[n]\alpha_{n-1}}. \tag{3.23}$$

In (3.22) we recognise $\Gamma^{[1]}$ as a vector of size d_1, which is the size of the bond α_1. The operation of summing over α_1, the result of which is placed in brackets in (3.23), can thus be described as multiplying the vector $\Gamma^{[1]}$ by the matrix $\Gamma^{[2]}$. The result is a vector of size d_2, the size of the bond α_2. Summing over α_2 will require another vector-matrix operation, and so on until the coefficient is obtained. This procedure of summing over the bonds one by one and each time obtaining a new tensor is called contraction. The first three steps of the contraction of a five qubit state is shown in figure 3.2, where we denote selecting an index i for each tensor by a series of red squares. It is also evident from this diagram that the contraction process amounts to multiplying a series of vectors by matrices.

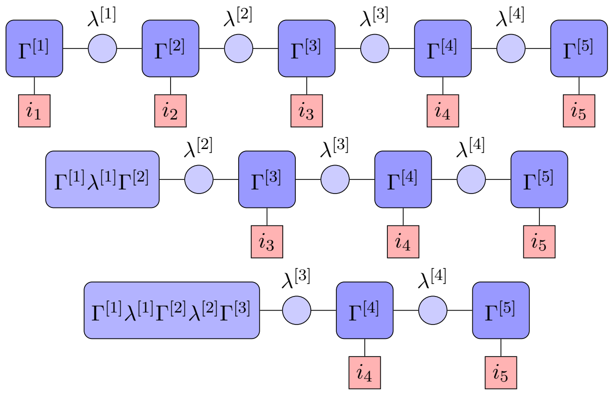


Figure 3.2: The first three steps in the contraction of a five qubit MPS.

The time required to sum over an index is the on order of the product of the sizes of each index in the two tensors involved which are not summed over, multiplied by the size of the summed index. The operation in brackets in (3.23) thus takes time $O(d_1 d_2)$, where d_1 is the size of the ancillary dimension α_1 and d_2 is the size of the ancillary dimension α_2. The following operation, summing over α_2, will take time $O(d_2 d_3)$ where d_2 and d_3 are defined similarly to d_1 and d_2. The total time required to obtain a coefficient in this state is thus $O\left(\sum_{i=1}^{n-2} \alpha_i \alpha_{i+1}\right)$. If most of the bonds in a state have the same dimension of d, this gives a time required to obtain a coefficient of $O(nd^2)$, where n is the number of qubits.

Operators

It is important when using an MPS for simulations in quantum computing that the representation can be easily updated after applying quantum operators. The efficiency of applying operators with the MPS form (3.4) was shown in [40].

If we have a MPS representation and want to apply a one qubit gate $\mathbf{U} = \sum_{a,b} U_{a,b}|a\rangle\langle b|$ to qubit k, we have:

$$\begin{aligned}\mathbf{U}|\psi\rangle = &\sum_{a,b}\sum_{i_1,\ldots,i_n}\sum_{\alpha_1,\ldots,\alpha_{n-1}} U_{a,b}\Gamma^{[1]\alpha_1}_{i_1}\ldots\lambda^{[k-1]}_{\alpha_{k-1}}\Gamma^{[k]\alpha_{k-1}\alpha_k}_{i_k}\lambda^{[k]}_{\alpha_k}\ldots\Gamma^{[n]\alpha_{n-1}}_{i_n} \\ &\qquad |i_1,\ldots,i_{k-1}\rangle\otimes|a\rangle\langle b|i_k\rangle\otimes|i_{k+1},\ldots,i_n\rangle \qquad (3.24)\\ = &\sum_{i_1,\ldots,i_n}\sum_{\alpha_1,\ldots,\alpha_n}\Gamma^{[1]\alpha_1}_{i_1}\ldots\lambda^{[k-1]}_{\alpha_{k-1}}\Gamma'^{[k]\alpha_{k-1}\alpha_k}_{i_k}\lambda^{[k]}_{\alpha_k} \\ &\qquad \ldots\Gamma^{[n]\alpha_{n-1}}_{i_n}|i_1,\ldots,i_k,\ldots,i_n\rangle, \qquad (3.25)\end{aligned}$$

where $\Gamma'^{[k]\alpha_{k-1}\alpha_k}_{i_k} = \sum_a U_{i_k,a}\Gamma^{[k]\alpha_{k-1}\alpha_k}_a$. This procedure is shown in figure (3.3) for a small region of a larger MPS. Only a single tensor is being changed, and so this application takes time $O(d_1 d_2)$ where d_1 and d_2 are the bond dimensions on the left and right of tensor k respectively.

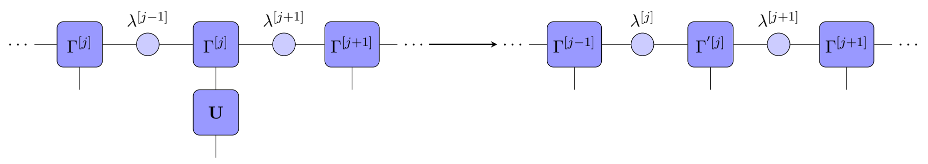


Figure 3.3: The application of a single qubit operator $\mathbf{U}$ to a MPS.

The application of a two qubit gate to a MPS is slightly more complicated. To multiply the gate by the state on qubits $j, j+1$, the tensors corresponding to these two qubits must be contracted:

$$\Gamma^{[j]\alpha_{j-1}\alpha_j}_{i_j}\lambda^{[j]}_{\alpha_j}\Gamma^{[j+1]\alpha_j\alpha_{j+1}}_{i_{j+1}} \rightarrow A^{\alpha_{j-1}\alpha_{j+1}}_{i_j i_{j+1}}. \qquad (3.26)$$

Once this has occurred the new combined tensor can be updated with the gate $\mathbf{U} = \sum_{i_1,j_1,i_2,j_2} U_{i_1,i_2,j_1,j_2}|j_1,j_2\rangle\langle i_1,i_2|$:

$$A^{\alpha_{j-1}\alpha_{j+1}}_{i_j i_{j+1}} \rightarrow A'^{\alpha_{j-1}\alpha_{j+1}}_{i_j i_{j+1}} = \sum_{ab} A^{\alpha_{j-1}\alpha_{j+1}}_{ab} U_{a,b,i_j,i_{j+1}}. \qquad (3.27)$$

Finally, the updated tensor $A'^{\alpha_{j-1}\alpha_{j+1}}_{i_j i_{j+1}}$ is decomposed via a SVD into a new set of tensors and a vector:

$$A'^{\alpha_{j-1}\alpha_{j+1}}_{i_j i_{j+1}} \to \Gamma'^{[j]\alpha_{j-1}\alpha_j}_{i_j} \lambda'^{[j]}_{\alpha_j} \Gamma'^{[j+1]\alpha_j\alpha_{j-1}}_{i_{j+1}}. \tag{3.28}$$

This procedure is shown in figure 3.4.

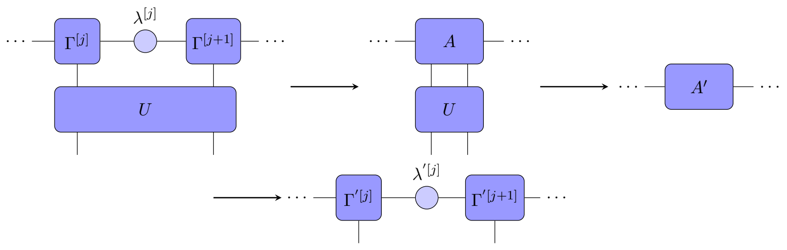


Figure 3.4: The application of a two qubit gate to a MPS.

The most time consuming steps in this operation are the initial contraction of two adjacent tensors and the SVD. The first of these steps takes time $O(d_1 d_2 d_3)$ where d_1, d_2 and d_3 are the left, middle and right bonds in the two tensor system. The second step takes time $O(\max(d_1, d_2)\min(d_1, d_2)^2)$. As such, if the bonds have a common size of χ, applying a two qubit gate to a MPS will take time $O(\chi^3)$.

Similarity between different MPSs

The question naturally arises when dealing with MPSs whether two MPSs encode the same quantum state. If both MPSs are in the canonical representation, it is straightforward to compare the tensors and vectors one by one. If each tensor or vector in one MPS is the same (up to computational accuracy) as the corresponding tensor or vector in the other MPS, then the two states are the same.

However, if one or both of the MPSs is not in the canonical gauge, the MPSs may encode the same state if their constituent tensors and vectors are different and if their bond dimensions are different. In this case, it is possible to compare each coefficient of the MPS individually by contraction. However, the time required to do this scales exponentially with the size of the system and this does not exploit the properties of the MPS representation which make it beneficial for simulation. A better approach is to compute a distance metric between the two MPS quantum states being considered.

The easiest distance metric to compute is the inner product between the two states. As MPSs encode pure states, this is equivalent to calculating the fidelity between the two states.

We will consider how to calculate the inner product between two states:

$$
\begin{aligned}
|\psi\rangle &= \sum_{\alpha_1,\ldots,\alpha_n} \sum_{i_1,\ldots i_n} \Gamma^{[1]\alpha_1} \lambda^{[1]}_{\alpha_1} \ldots \Gamma^{[n]\alpha_{n-1}}_{i_n} |i_1,\ldots,i_n\rangle \\
|\phi\rangle &= \sum_{\beta_1,\ldots,\beta_n} \sum_{j_1,\ldots j_n} \Theta^{[1]\alpha_1} \upsilon^{[1]}_{\beta_1} \ldots \Theta^{[n]\beta_{n-1}}_{i_n} |j_1,\ldots,j_n\rangle .
\end{aligned}
$$

Taking the inner product $\langle\psi|\phi\rangle$ will cause a product $\langle i_m|j_m\rangle = \delta_{i_m j_m}$ for each qubit m, and so we need to contract over the indices $\{i_m\}$ and the corresponding $\{j_m\}$ to form the inner product. Together with the sums over the ancillary indices, this gives us a tensor network to contract to calculate the inner product, shown in figure 3.5. The most efficient contraction pattern

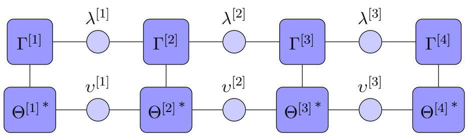


Figure 3.5: A tensor network to calculate the inner product of two MPSs.

will depend upon the dimenions of the bonds. A generally efficient contraction can be performed by moving from left to right using a pattern shown in figure 3.6. As the tensor being contracted into only ever has two bonds to neighbouring tensors (as well as a quantum state index in common), this contraction will take time $O(n \min(d_1, d_2) \max(d_1, d_2)^2)$ where d_1 and d_2 are the maximum bond sizes of $|\psi\rangle$ and $|\phi\rangle$ respectively and n is the number of qubits (which much be the same in both states).

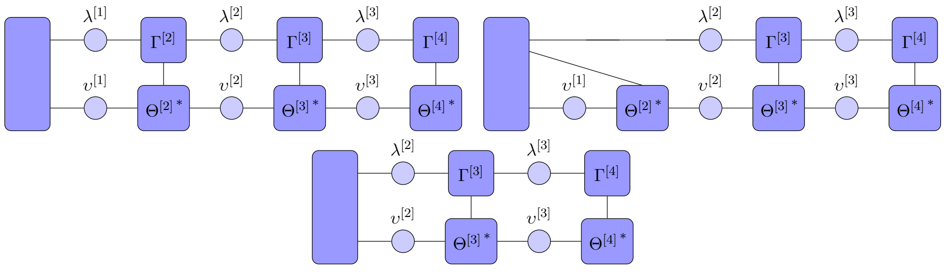


Figure 3.6: The first three steps of the efficient contraction of the inner product of two MPSs.

Expectation Values

Given a quantum state represented as a MPS, we often want to compute the expected value of an operator applied to the state. To show how this calculation can be performed, we first consider the calculation of the expectation value of a two qubit operator $\langle\hat{U}\rangle = \langle\psi|\hat{U}|\psi\rangle$. We can regard $|\phi\rangle = \hat{U}|\psi\rangle$, and so $\langle\hat{U}\rangle = \langle\psi|\phi\rangle$. As such, the computation of the expected value of $\hat{U}$ involves applying $\hat{U}$ to a copy of $|\psi\rangle$, and then taking an inner product with $|\psi\rangle$. The first of these operations has cost $O(\chi^3)$, where χ is the maximal Schmidt rank of $|\psi\rangle$, and the second operation has cost $O(n\chi^3)$. This gives an overall cost for the calculation of $O(n\chi^3)$.

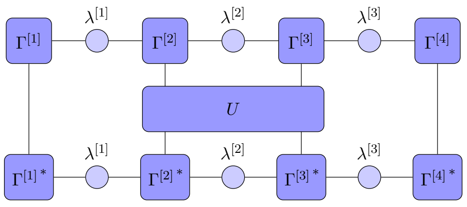


Figure 3.7: A tensor network to calculate the expectation value of a two qubit unitary on a four qubit MPS.

We can regard larger unitary operators as sums and products of smaller one and two qubit unitary operators. It is then clear that to calculate the expected value of these operators you apply them to a state $|\psi\rangle$ and then take an inner product with itself. As long as the quantum circuit whose expectation value you are finding is of logarithmic depth and the value of χ is small, this may be performed in an amount of time growing polynomially with the number of qubits.

Density Matrices

To show how to calculate a one qubit density matrix of a qubit in a MPS, we consider the Schmidt decomposition of a state

$$|\psi\rangle = \sum_i \lambda_i |a_i\rangle|b_i\rangle. \tag{3.29}$$

If we label the bipartition over which this decomposition is created as $A|B$, then it is straightforward to calculate the density matrices ρ_a, ρ_b by using the Schmidt decomposition (3.9):

$$\rho_a = \sum_i \lambda_i^2 |a_i\rangle\langle a_i|, \quad \rho_b = \sum_i \lambda_i^2 |b_i\rangle\langle b_i|. \tag{3.30}$$

And so we can see that if we are given the Schmidt decomposition of a state, the density matrix of one part of the state does not depend on Schmidt vectors of the other part of the state, only on the Schmidt coefficients and the Schmidt vectors in the required subspace. Note that if we are given any other decomposition of the state, then this is not in general true if the vectors on each side of the decomposition are not orthogonal. As the canonical form of the MPS specifies a state in terms of the Schmidt decomposition over each bipartition, this means that the reduced density matrix from each qubit depends only upon its local tensor and the vectors adjacent to it. The tensor network required to calculate a one qubit density matrix is shown in figure 3.8. This operation takes time $O(\chi^2)$

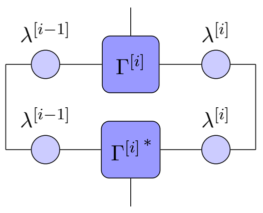


Figure 3.8: The tensor network to calculate the one qubit reduced density matrix of qubit i of a MPS in canonical form.

Density matrices of larger subsystems of a MPS can be calculated in a similar way. First, the required subsystem is contracted until it is represented by a single tensor. The density matrix is then formed from this tensor with exactly the same network as shown in figure 3.8, but the open quantum state indices will have be a larger size than two.

3.1.3 An analytic example

We give an example of a state which has an analytic MPS representation in order to further elucidate the operation of MPSs. The example we give is the GHZ state which is given by:

$$|\psi\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle^{\otimes N} + |1\rangle^{\otimes N}\right). \tag{3.31}$$

A version of this state with n qubits can easily written as a MPS with bond rank two, using a method similar to that which we will introduce in chapter 6.

$$\begin{aligned}\Gamma^{[0]\alpha_0}_{i_0} &= \delta_{\alpha_0 i_0}, \quad \Gamma^{[n]\alpha_{n-1}}_{i_n} = \delta_{\alpha_{n-1} i_n},\\ \lambda^{[0]}_{\alpha_0} &= \frac{1}{\sqrt{2}}, \lambda^{[m]}_{\alpha_m} = 1 \text{ for all } m \neq 0\\ \Gamma^{[m]\alpha_{m-1}\alpha_m}_{i_m} &= \delta_{\alpha_{m-1}\alpha_m}\delta_{\alpha_m i_m} \text{ for all } m \neq \{0, n\}\end{aligned} \tag{3.32}$$

We can see from this example that this state has non-trivial entanglement but it is easy to represent and perform calculations on with a MP3 representation. We note that it is also easy to represent this state without a MPS as it only has two non-zero coefficients in the computational basis.

3.2 Projected entangled pair states

It is possible to generalise the MPS representations in several ways. One natural generalisation is increase the number of ancillary indices connecting to each tensor, allowing states with 2D, 3D and other higher dimensional couplings to be represented more efficiently. Such a tensor network is called a projected entangled pair state (PEPS) [64]. We show a pictorial representation of a PEPS state with 16 quoits in two dimensions in figure 3.9 In this thesis we will confine our simulations to one dimension, but we give an overview of the properties of PEP states for completeness.

Some of the properties of MPSs also apply to PEPS. For example, by allowing the bond sizes to grow exponentially with the size of the quantum system any quantum state can be represented. However, the increased dimensionality of PEPSs causes some of their properties to differ from those of MPSs. A notable difference is the difference in the computational complexity of the two state representations. While MPSs may be contracted efficiently, exact contraction of PEPSs is #P-complete [119]. In general, there is no order in which the indices of a PEPS may be contracted for which the computation time of the contraction will not scale exponentially in the number of constituents of the quantum system. This difficulty extends to the computation of observables, which is straightforward in a MPS but inefficient in a PEPS. Contractions in PEPSs must then be carried out with approximations. For example, in [64] a scheme is presented in which

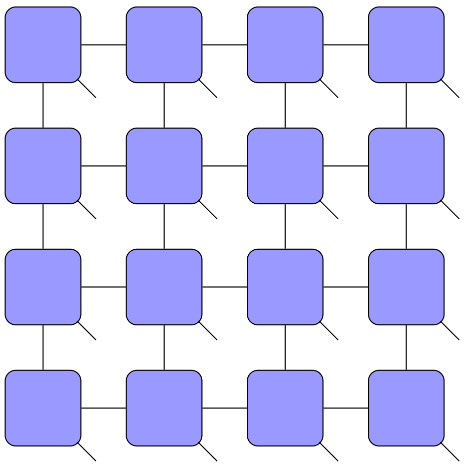


Figure 3.9: A pictorial representation of a PEPS in two dimensions with 16 qubits.

a 2D PEPS is contracted by regarding the top row of the lattice as a MPS and each remaining row as a MPO. The MPOs are contracted into the MPS one by one, and each time the bonds in the MPS are truncated to stop its size growing too large. A different approximate contraction scheme can be found in [120].

3.3 Matrix product operators

A generalisation of the MPS formalism is to keep the linear connectivity of the tensor network but to encode an operator rather than a state. Doing so, an operator $\hat{O}$ may be written as

$$\hat{O} = \sum_{\alpha_1,\ldots,\alpha_n} \sum_{i_1,j_1,\ldots,i_n,j_n} \Theta^{[1]\alpha_1}_{i_1 j_1} \gamma^{[1]}_{i_1} \Theta^{[2]\alpha_1\alpha_2}_{i_2 j_2} \ldots \Theta^{[n-1]\alpha_{n-2}\alpha_{n-1}} \gamma^{[n-1]}_{\alpha_{n-1}} \Theta^{[n]\alpha_{n-1}}_{i_{n-1} j_{n-1}} |j_1,\ldots,j_n\rangle\langle i_1,\ldots,i_n|. \tag{3.33}$$

The resulting representation is called a matrix product operator (MPO). A pictorial representation of a MPO is shown in figure 3.10. These operators were introduced in [65, 66] in the context of extending DMRG to mixed states. If a density matrix is written in this format, it is known as a matrix product density operator (MPDO). MPOs have been used, for example, to perform time evolution on MPSs [67, 68] and to study non-equilibrium steady states [121]. As such, a large amount of work has been done on which kinds of Hamiltonians, or the exponentiation of Hamiltonians, can be expressed efficiently as a MPO [67]. In particular, [122] presents an explicit method of constructing MPOs for arbitrary Hamiltonians and shows the representation to always be efficient for Hamiltonians composed of pairwise interactions. However, little work has been done on using MPOs to simulate quantum computing algorithms, the topic of this thesis.

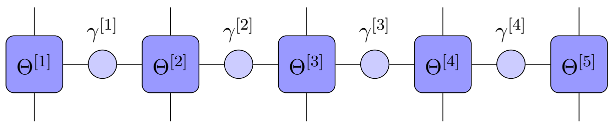


Figure 3.10: The pictorial representation of a five qubit MPO.

Similarly to MPSs, a canonical form can be found by applying a series of SVDs. Instead of being the Schmidt ranks of a state, the singular values are then the Schmidt numbers of the operator.

3.3.1 Calculations with MPOs

Many calculations in MPOs proceed in a similar way to the calculations used in MPSs, so we will be more brief in outlining them here.

One and two qubit gates

Similarly to applying one qubit gates to a MPS, applying a one qubit gate to a MPO at qubit number i involves updating only the tensor $\Theta^{[i]}$ located at site i. In contrast to the MPS case, there are two open indices per tensor in a MPO, one each for the input and output of the operator at each qudit. As such, the gate being applied can be multiplied into the top or bottom of the MPO. The choice of which way to do this multiplication depends upon the time ordering of the circuit or components of the operator in question: gates applied to the top of the MPO happen at a time before that of any yet applied gate while those applied to the bottom happen at a time later than any yet applied gate.

The application of a two qubit gate to a MPO also proceeds in an analogous way to the application of a two qubit gate to a MPS. The steps are:

1. Combine two adjacent tensors $\Theta^{[k]\alpha_{k-1}\alpha_k}_{i_k j_k}\gamma^{[k]}_{\alpha_k}\Theta^{[k+1]\alpha_k\alpha_{k+1}}_{i_{k+1}j_{k+1}}$ into a combined tensor $A^{\alpha_{k-1}\alpha_{k+1}}_{i_k j_k i_{k+1} j_{k+1}}$.

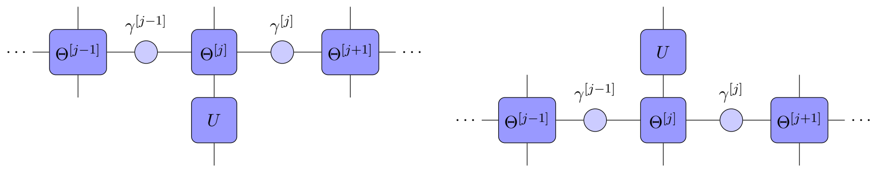


Figure 3.11: The application of a single qubit gate **U** to a MPO at the top and bottom of the operator respectively.

2. Multiply the combined tensor by the gate being used. Here multiplying above or below corresponds to contracting the i_k, i_{k+1} and j_k, j_{k+1} indices respectively.

3. Perform a SVD to split the new combined tensor into new individual tensors $\Theta'^{[k]\alpha_{k-1}\alpha_k}_{i_kj_k}\gamma'^{[k]}_{\alpha_k}\Theta'^{[k+1]\alpha_k\alpha_{k+1}}_{i_{k+1}j_{k+1}}$.

Applying a MPO to a MPS

One of the reasons that MPOs are useful in simulating quantum systems is that they can be directly applied to MPSs to simulate the action of a given operator on a given state. This application produces a new MPS in which each tensor is the product of a tensor in the original MPS and a tensor in the MPO, each representing the same qubit. That is, for each l,

$$\Gamma'^{[l]\beta_{l-1}\beta_l}_{i_l} = \Gamma'^{[l]\alpha_{l-1}\kappa_{l-1}\alpha_l\kappa_l}_{i_l} = \sum_{j_l} \Gamma^{[l]\alpha_{l-1}\alpha_l}_{j_l}\Theta^{[l]\kappa_{l-1}\kappa_l}_{j_li_l}. \tag{3.34}$$

where we have relabelled the ancillary indices to include those from both the state and the operator. Similarly the vectors are transformed as

$$\lambda'^{[l]}_{\beta_l} = \lambda'^{[l]}_{\alpha_l\kappa_l} = \lambda^{[l]}_{\alpha_l}\gamma^{[l]}_{\kappa_l}. \tag{3.35}$$

This multiplication is shown pictorially in figure 3.11.

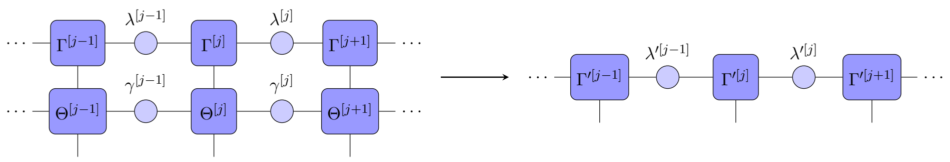


Figure 3.12: The application of a MPO to a MPS.

For each bond in the MPS with a bond size of χ_s and a corresponding bond size in the MPO of χ_o, the new MPS will have a bond size immediately after the multiplication of $\chi_s\chi_o$, and so the affect of applying a MPO is a multiplicative increase in the bond sizes of the MPS. The new MPS formed may not be in canonical form, although it can be returned to this form by a series of SVDs along each affected bond. As such, the new Schmidt rank of the quantum state may be less than $\chi_s\chi_o$. Overall, the application of a MPO to a MPS takes time $O(n\xi_o^2\xi_s^2)$ where ξ_o and ξ_s are the maximal bond sizes of the MPO and MPS respectively.

The rank increase of the new MPS before being returned to canonical form invites an interpretation of the Schmidt number of an operator as an upper bound for the amount by which the Schmidt rank of a state can increase upon application of the operator. If the MPO has bond rank d for that bipartition, the Schmidt rank will be at most multiplied by d. However, as stated above, in many cases the Schmidt rank after application will be much lower than this.

Applying a MPO to a MPO

Analogously to the application of a MPS to a MPO, it is possible to apply one MPO to another in order to create a new operator. The second MPO may be any size, and so it is possible to apply a two qubit gate to a MPO, for example, by determining its MPO form and then performing a series of contractions between two MPOs. The resulting MPO will be in non-canonical form and have bond dimensions which are the bond dimensions of each of the product operators multiplied. The new operator can be returned to canonical form by performing a series of SVDs. As with the application of a gate to a MPS, a MPO may be applied to the top or bottom of another MPO, depending on which operator takes place first physically.

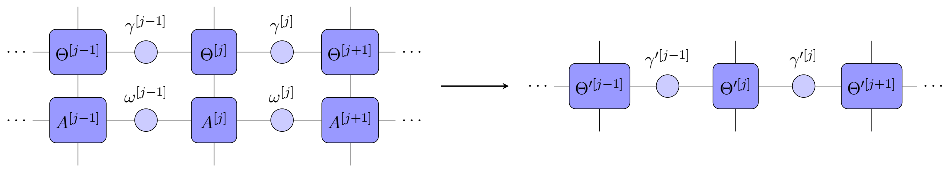


Figure 3.13: The application of a MPO to another MPO.

Adding MPOs

Operators are often written as a sum of other simpler operators. As such, it is useful to be able to construct unknown MPOs by adding the MPO representations of other operators whose MPO representations is known. This is also necessary, for example, to perform a quantum operation which is expressed as a set of Krauss operators on a density matrix. Tensor multiplication tells us how to construct an operator as a product of other operators, but not as a sum.

A straightforward way to add MPOs is to consider the linearity of operator actions: $\hat{U}|\psi\rangle = \left(\hat{U_1} + \hat{U_2}\right)|\psi\rangle = \hat{U_1}|\psi\rangle + \hat{U_2}|\psi\rangle$. As such, we may construct a tensor network which is the sum of two MPOs by introducing a new index specifying which operator is being used at any one time:

$$\begin{aligned}\hat{U} = &\left(\sum_{\alpha_1\ldots\alpha_{n-1}} \sum_{i_1\ldots i_n j_1\ldots j_n} A^{[1]\alpha_1}_{i_1 j_1} \lambda^{[1]}_{\alpha_1} \ldots \lambda^{[n-1]}_{\alpha_{n-1}}\right.\\ &\qquad\left. A^{[n]\alpha_{n-1}}_{i_n j_n} |i_1 \ldots i_n\rangle\langle j_1 \ldots j_n|\right)\\ &+\left(\sum_{\beta_1,\ldots,\beta_{n-1}} \sum_{i_1,j_1,\ldots,i_n,j_n} B^{[1]\beta_1}_{i_1 j_1} \upsilon^{[1]}_{\beta_1} \ldots \upsilon^{[n-1]}_{\beta_{n-1}}\right.\\ &\qquad\left. B^{[n]\beta_{n-1}}_{i_n j_n} |j_1 \ldots j_n\rangle\langle i_1 \ldots i_n|\right) \qquad (3.36)\\ = &\sum_{o=1}^{2} \sum_{\alpha_1\ldots\alpha_{n-1}=1} \sum_{i_1\ldots i_n j_1\ldots j_n} \Theta^{[1]o\,\alpha_1}_{i_1 j_1} \gamma^{[1]o}_{\alpha_1} \ldots \gamma^{[n-1]o}_{\alpha_{n-1}}\\ &\qquad \Theta^{[n]o\,\alpha_{n-1}}_{i_n j_n} |j_1 \ldots j_n\rangle\langle i_1 \ldots i_n|, \qquad (3.37)\end{aligned}$$

where we have relabeled $A^{[k]\alpha_{k-1}\alpha_k}_{i_k j_k} = \Theta^{[k]1\,\alpha_{k-1}\alpha_k}_{i_k j_k}$, $B^{[k]\beta_{k-1}\beta_k}_{i_k j_k} = \Theta^{[k]2\,\alpha_{k-1}\alpha_k}_{i_k j_k}$, $\lambda^{[k]}_{\alpha_k} = \gamma^{[k]1}_{\alpha_k}$ and $\upsilon^{[k]}_{\beta_k} = \gamma^{[k]2}_{\alpha_k}$ and we have relabeled $\beta_k = \alpha_k$ for the tensors B and the vectors υ. In order for the addition of this extra index to be correct, we have to change the upper bound of each α_k to be $\max(\text{size}(\alpha_k), \text{size}(\beta_k))$, and set to zero any extra elements created in this process.

In a similar process to the relabeling of the ancillary indices when applying an MPO to a MPS or MPO, we can relabel the ancillary indices, including o, in each tensor and vector in (3.37). That is, if the size of bond

k is d_k, set $\alpha'_k = o\, d_k + \alpha_k$. Doing so allows us to rewrite (3.37) as:

$$\hat{U} = \sum_{\alpha'_1 \ldots \alpha'_{n-1}=1} \sum_{i_1 \ldots i_n j_1 \ldots j_n} \Theta^{[1]\alpha'_1}_{i_1 j_1} \gamma^{[1]}_{\alpha'_1} \cdots \gamma^{[n-1]}_{\alpha'_{n-1}} \Theta^{[n]\alpha'_{n-1}}_{i_n j_n} |j_1 \ldots j_n\rangle\langle i_1 \ldots i_n|, \tag{3.38}$$

and so we have constructed a MPO which sums two constituent MPOs. It will not be in the canonical form and will have a bond size of $d^1_k + d^2_k$ for each k, where d^1_k and d^2_k are the sizes of the kth bond in the first and second MPOs respectively. The new MPO can be placed back in the canonical form by doing a series of SVDs across each bond moving from left to right and then again from right to left. The second series of SVDs is needed to ensure that information from each tensor and bond reaches each other tensor and bond.

To clarify the above method, we will provide an alternate explanation. We consider each tensor as a matrix of one qubit operators, each of which is not necessarily unitary. The location of each operator in this matrix is given by its left and right bond indices:

$$A^{[k]} = \begin{pmatrix} \mathbf{A}^{[k]1\,1} & \ldots & \mathbf{A}^{[k]1\,d} \\ \vdots & \ddots & \vdots \\ \mathbf{A}^{[k]d\,1} & \ldots & \mathbf{A}^{[k]d\,d} \end{pmatrix}. \tag{3.39}$$

We are free to increase the size of any such matrix as long as we set any added element to zero. If we have two MPOs, which at each site k have left and right bond dimensions l_1, r_1 and l_2, r_2 respectively, we are thus free to add additional unused elements to make the dimensions of the matrices

$(l_1 + l_2, r_1 + r_2)$.

$$\mathbf{A}'^{[k]} = \begin{pmatrix} \mathbf{A}^{[k]1\,1} & \dots & \mathbf{A}^{[k]1\,d} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{[k]d\,1} & \dots & \mathbf{A}^{[k]d\,d} & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix} \tag{3.40}$$

$$\mathbf{B}'^{[k]} = \begin{pmatrix} 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \mathbf{B}^{[k]1\,1} & \dots & \mathbf{B}^{[k]1\,d} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & \mathbf{B}^{[k]d\,1} & \dots & \mathbf{B}^{[k]d\,d} \end{pmatrix} \tag{3.41}$$

If we add these two matrices for every set of corresponding tensors in each MPO we will then have a new MPO whose action is the sum of the actions of the product MPOs.

$$\begin{pmatrix} \mathbf{A}^{[k]1\,1} & \dots & \mathbf{A}^{[k]1\,d_1} & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}^{[k]d_1\,1} & \dots & \mathbf{A}^{[k]d_1\,d_1} & 0 & \dots & 0 \\ 0 & \dots & 0 & \mathbf{B}^{[k]1\,1} & \dots & \mathbf{B}^{[k]1\,d_2} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \ddots \\ 0 & \dots & 0 & \mathbf{B}^{[k]d_2\,1} & \dots & \mathbf{B}^{[k]d_2\,d_2} \end{pmatrix}. \tag{3.42}$$

We can see from (3.42) that in effect, we can add two MPOs by adding each of their tensors in block diagonal fashion. This can be extended to an arbitrary number of MPOs, which can be added by adding the sets of corresponding tensors in a block diagonal fashion, as long as consistent index numbering rules are used.

3.4 Quantum Circuit Simulation

Quantum circuits can be simulated with a MPS by starting with a desired input MPS and applying the required gates in order. Most quantum algorithms use a product state as input (many use a computational basis state), and so it is easy to generate this input state for the simulation. At each time step, each required one qubit quantum gate can be applied directly to the MPS and each two or higher qubit gate can be applied by the method specified in section 3.3.1. After applying the gate, it is optional whether or not to decompose the combined quantum system the gate has applied to. Instead, at any time two or more adjacent qubits can be combined into a qudit and considered together. If a composite qubit system is decomposed after applying a two qubit gate, it can be decomposed with several different methods:

- A SVD may be used. This is the method that has been described above in sections 3.3.1.
- Another decomposition such as the QR decomposition may be used. Both rank revealing and non-rank revealing decompositions may be used. Using such a decomposition will be beneficial if the computational resources required to perform it are less than those for the SVD.
- If the bond rank is unimportant or it may be surmised from the quantum circuit that it will be the maximum allowable rank given the quantum dimensions of the qubits, a trivial decomposition may be used:

$$A_{m\times n} = \begin{cases} A_{m\times n} I_{n\times n}, & m \geq n \\ I_{m\times m} A_{m\times n}, & m < n \end{cases} \tag{3.43}$$

Performing a quantum circuit simulation with a MPO requires slightly different techniques. A quantum circuit simulation will generally entail applying a quantum circuit to a particular input state. Such a simulation could involve the use of MPSs and MPOs to increase the speed of the simulation. A decision must then be made when setting up the simulation which parts of the quantum circuit will be contracted into a MPO and which parts will be applied to a given MPS state each time the simulation is performed. Typically if multiple simulations are being performed, circuit sections which are common to each simulation should be contracted into MPOs. However the resulting MPO may become highly entangling while the MPS is only slightly entangled throughout the simulation. In this case contracting part of the circuit into a MPO will slow down a simulation. In this way the entangling properties of a quantum circuit are important in the determination of whether forming a MPO will be a worthwhile exercise. There may also be utility in forming a MPO of a circuit or algorithm to explicitly study the properties of the MPO. By doing so the entangling properties of the algorithm may be studied. We perform such analyses in chapters 4 and 6.

The initial state of a circuit simulation with a MPO is the identity matrix on the required number of qubits. The n qubit identity matrix may be encoded in a MPO by noting that it is separable and so can be decomposed into n one qubit identity operators:

$$I^{[n]} = I^{\otimes N} = I_1 \otimes \ldots \otimes I_n. \tag{3.44}$$

This identity MPO may be regarded as being situated at any time slice within a quantum circuit. A time slice would pictorially be represented as a vertical line cutting through a quantum circuit diagram. Any gates after this time slice must be applied in order to the output indices of the MPO. Similarly any quantum gates happening before this time slice must be applied in reverse order to the input indices of the MPO. As long as each of these orderings are preserved, no ordering is required between when gates are applied to the input indices and when gates are applied to the output indices. MPO simulations may also entail adding two (not necessarily unitary) MPOs to produce a larger MPO of a unitary operator as well as applying MPOs to other MPOs instead of applying gates to MPOs. Our statements about time orderings apply in these cases as well. That is, a MPO of part of a circuit may be applied to the input or output indices of another part of the circuit as long as the overall time ordering is preserved.

3.5 Entanglement in matrix product states

Entanglement may be considered as a quantum correlation existing between two different parts of a quantum system. MPSs are MPOs are a way of writing quantum states and operators in a way in which all information is presented locally with the bonds representing quantum and classical correlations. As such, MPSs and MPOs present natural encodings of quantum states and operators if information about entanglement and correlations is required.

MPSs are written in a way that elucidates the amount of entanglement is a system after few or no calculations. The Schmidt rank of a system along a particular bipartition provides a coarsely grained measurement of the amount of entanglement between two halves of a system. The Schmidt rank is the bond rank of a MPS and so there is a direct connection between the size of the bonds and the amount of entanglement in the corresponding state. A set of Schmidt coefficients of a state $\{c_i\}$ directly translates into eigenvalues of the reduced density matrix $\{|c_i|^2\}$ after tracing out the subsystem corresponding to either half of the bipartition. As such, the entropy of entanglement of a MPS along a bipartition may be calculated from the vector of coefficients on the corresponding bond $\lambda^{[m]}_{\alpha_m}$ provided the MPS is in the canonical gauge:

$$S = -\sum_{\alpha_m} |\lambda^{[m]}_{\alpha_m}|^2 \log |\lambda^{[m]}_{\alpha_m}|^2. \tag{3.45}$$

While the Schmidt rank of a state is a measure of the amount of entanglement in the state, the Schmidt number of an operator does not have as straight-forward an interpretation. An operator with a high Schmidt number may have a high amount of classical correlating power but little entangling ability. This is illustrated in the case of two-qubit unitary operations by the SWAP gate, which has the Schmidt-operator decomposition $1/2(I \otimes I + X \otimes X + Y \otimes Y + Z \otimes Z)$. The Schmidt number of four is the maximum possible for a two-qubit operator, but the gate has no entangling power. As such, if a MPO has high bond ranks it is not necessarily a highly entangling operator. As noted in section 3.3.1, an interpretation of the Schmidt number of an operator is an upper bound on the amount by which the Schmidt rank of a state will increase upon application of the operator.

There are two measures of correlating ability which can be directly calculated from a MPO. Firstly, the Hartley Strength [123] of an operator is given by $\log_2(s)$ where s is the maximal Schmidt number of the operator. The maximal Schmidt number of an operator is the maximal bond rank of the MPO in the canonical gauge. The singular values $\{s_i\}$ of an operator along any bipartition allow a probability distribution to be derived given by

$p_i = s_i^2/D$ where D is the dimension of the Hilbert space the operator acts in. The second measure of correlating ability is the Schmidt strength, which is defined to be the Shannon entropy of this probability distribution [123]:

$$K_{\text{sch}}(U) = -\sum_i \left(\frac{s_i^2}{D}\right) \log_2 \left(\frac{s_i^2}{D}\right). \tag{3.46}$$

The Schmidt strength is the maximum entropy of the probability distribution following from the singular values of an operator U along any bipartition. It also gives the maximum entropy $E(U|\alpha\rangle|\beta\rangle)$ where $|\alpha\rangle$ and $|\beta\rangle$ are states in two different quantum systems corresponding to any bipartition of the operator U. The states $|\alpha\rangle$ and $|\beta\rangle$ are maximally entangled with ancillary systems. As the singular values of an operator along bipartition are contained in the bond vectors in a MPO, the calculation of (3.46) along any bipartition is straightforward given the canonical MPO decomposition of the operator in question.

3.5.1 Area Laws

The popularity of MPS representations in condensed matter calculations is grounded in the connection between the size of the MPS representation of a state and the amount of entanglement in the state. While this connection allows for efficient classical simulation of quantum states with little entanglement, it also makes the MPS representation a good basis for an ansatz of a condensed matter ground state. Many physical states arise from local interactions and so the ground states of these systems are typically less entangled than most of the states found in their Hilbert spaces. This notion is made precise through an area law. That is, given two regions, the entanglement entropy between them for many Hamiltonians scales linearly (with a possible logarithmic correction) with the area of the interface, rather than the volumes, of the regions. For a one-dimensional system, the interface between two halves of a system is a single point. As such, one-dimensional states which obey an area law are precisely those whose ground states can be represented efficiently with a MPS. A MPS will thus provide a very good variational ansatz for determining the ground state of a given Hamiltonian to which an area law applies. This is the reason for the success of density matrix renormalisation group techniques, which can be considered as performing variational minimisation with a tensor network ansatz [124]. Similar statements can be made about two dimensional systems, in which PEPS representations are effective encodings of states which obey an area law. There are many results concerning which systems an area law apply to, but this topic is not of primary relevance to this thesis. An introduction to and review of area law results may be found in [125].

Chapter 4

Simulation of the quantum Fourier transform

In this chapter, we will numerically represent the quantum Fourier transform (QFT) in the matrix product operator format. We will principally be concerned with the complexity of such a representation and the implications of this for the numerical difficulty of performing a simulation. We find that the number of elements in each operator bond grows as $O(\log n)$ as the number of qubits n increases, and so simulation of the QFT applied to a MPS with Schmidt rank χ takes $O(n(\log n)^2\chi^2)$ time. Section 4.1 will summarise existing classical simulation results for the QFT and section 4.2 will outline our approach. Sections 4.3 and 4.4 will present out results with respect to the time required for QFT simulation and its accuracy. Sections 4.6, 4.5 and 4.7 will discuss the reasons for this scaling and its implications.

4.1 Previous QFT simulation results

The QFT is the most intuitively quantum mechanical part of Shor's algorithm. That is, it contains Hadamard and controlled phase-rotation gates, neither of which have a classical analogue. The full QFT with an arbitrary input state does not display any of the features found in previous studies to allow efficient classical simulation. Despite this, the approximate QFT (AQFT) is efficiently classically simulatable for input states with limited entanglement. This was first shown in [126, 127] using a tensor contraction simulation method. Together with results showing that the AQFT is sufficient for many computational tasks including Shor's algorithm [100, 128, 129, 130], this result is sufficient to show that the QFT is efficiently classically simulatable to high fidelity for a limited class of input states. It was shown in [131] using matrix product states that a terminating QFT can be efficiently simulated for any input state with limited entanglement. Additionally, in [132] a classical algorithm to obtain the results of the QFT on separable states is derived. That this algorithm is simpler than the QFT suggests that the quantum speedup of the QFT lies in the quantum parallelism of its input state rather than its innate complexity.

A more general result is presented in [71, 72], where it is shown that a normalizer circuit can be simulated in polynomial time with a restriced set of inputs over an arbitrary abelian group:

$$G = \mathbf{Z}_{d_1} \times \cdots \times \mathbf{Z}_{d_m}, \tag{4.1}$$

where each $\mathbf{Z}_{d_i}$ is a cyclic group formed from integer addition modulo i. In this group, a normalizer circuit is defined to be one containing a polynomially deep set of operations, including adaptive operations conditioned on measurement:

- Quantum Fourier transforms on any $\mathbf{Z}_{d_i}$:

$$F_i = \frac{1}{\sqrt{d_i}} \sum e^{2\pi ixy/d_i} |x\rangle\langle y|. \tag{4.2}$$

- Gates formed from automorphisms on the group G. These correspond to unitary permutations on the standard computational basis.

- Quadratic phase gates, which correspond to applying a complex phase to every element $g \in G$.

These results generalize the Gottesman-Knill theorem and allow the efficient classical simulation of a circuit containing a polynomial number of QFTs as well as other gates which can generate large amounts of entanglement. Additionally, the resetricted set of input states in [71, 72] are much broader than in earlier work.

In this chapter, we use matrix product operators to simulate the quantum Fourier transform. Our numerical results imply that for weakly entangled input states over n qubits, the resources required for this simulation scale as

$O(n\,(\log(n))^2)$, a significant improvement on earlier results. The simulation tools we use are also more straightforward than tensor contraction methods such as the ones used in [126, 127], and can easily be applied to many different states.

4.2 Circuit and simulation methods

As described in section 2.4, the quantum Fourier transform (QFT) can be written in operator form:

$$\frac{1}{\sqrt{N}}\sum_{j,k=0}^{N-1} e^{2\pi ijk/N}|j\rangle\langle k|. \tag{4.3}$$

This equation can be expanded to give output values at individual qubits:

$$\begin{aligned}|j_1,\ldots,j_n\rangle \to \frac{1}{2^{n/2}}&\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\\ &\ldots\left(|0\rangle + e^{2\pi i 0.j_1\ldots j_n}|1\rangle\right),\end{aligned} \tag{4.4}$$

where $j = j_1 2^{n-1} + j_2 2^{n-2} + \ldots + j_n 2^0$ and $0.j_l \ldots j_m = j_l/2 + \ldots + j_m/2^{m-l+1}$. The decomposition in turn motivates the canonical decomposition of the QFT into quantum gates, which is shown in figure 2.11.

It is not possible to simulate gates at arbitrary distance with a matrix product operator, and so to simulate the QFT we require a quantum circuit in which only qubits which are nearest neighbours interact. We follow the treatment in [91] to create such a circuit. The LNN circuit is shown in figure 2.12.

The operator-Schmidt decomposition of the QFT has been calculated exactly [133] and the maximal Schmidt number of a n qubit transform is 2^n. Additionally, all of the singular values are equal. Simulating the QFT using a MPO representation of (4.4) would thus entail an exponential scaling in terms of execution recourses as the number of qubits is increased.

From (4.4) we note that the output at the first qubit depends only upon the input value at the last qubit, the input at the second depends upon the output at the last two qubits and so on. As such, the operator displays similar classical correlations between the input and output values to those in our earlier swap gate example. These correlations are expensive to encode in a MPO. A simple re-ordering of the input or output qubit values (but not both) from equation (4.4) produces a more easily encoded operator:

$$|j_1, \ldots, j_n\rangle \to \frac{1}{2^{n/2}} \left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right) \ldots \left(|0\rangle + e^{2\pi i 0.j_n \ldots j_1}|1\rangle\right). \tag{4.5}$$

In (4.5) the output at the first qubit depends only upon the input at the first qubit, the output at the second qubit upon the input at the first two qubits and so on. This ordering thus requires less information to be communicated across bonds and can be encoded in a smaller MPO.

To construct a MPO encoding (4.5) one can construct a MPO of (4.4) and then apply the SWAP gates leading to the required ordering to only one side of the MPO. This has the disadvantage that the MPO for (4.4) must be calculated first, which is computationally intractable for large numbers of qubits. A better approach is to apply a swap gate to the input qubits whenever one is applied to the output qubits. This makes the ordering of the input qubits the same at all times as that of the output qubits. This approach also produces the required ordering and invites an interpretation that the resulting MPO contains only interesting correlations rather than expensive swap correlations.

4.3 Representation size scaling

We constructed MPOs encoding equation (4.5) with a simple nearest neighbour circuit [91]. The bond ranks of the MPO encoding (4.5) were much lower than those required to encode (4.4). Figure 4.1 shows the size of each element in a bond in the center of a MPO representing (4.5) for 24 qubits. We display the probability distribution derived from the singular values $p_i = s_i^2/D$ where s_i are singular values and D is the dimension of the Hilbert space (2^{24} in this case).

The sizes of the bond elements displays a characteristic drop-off from lower to higher rank. This characteristic was present regardless of the size of the MPO (MPOs with up to 50 qubits were tested). Initally the rate of decrease is slow, but it quickly becomes exponential with the bond rank included. The exponential decrease of bond element size halted at a probability of around 10^{-40} at quadruple precision (128 bits), which corresponds

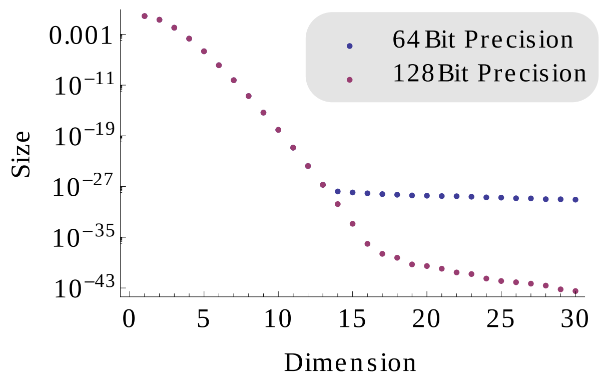


Figure 4.1: The probability distribution derived from the singular values of a bipartition at the center of MPOs representing the QFT with 24 qubits at two different precisions.

to a singular value of size relative to the largest value of 10^{-20}. There were many additional bond elements of this size or slightly smaller displaying a large amount of random variation in each MPO. While their size is much larger than machine precision (these results were produced for quadruple precision numbers with $\epsilon \approx 10^{-35}$), the condition number of a singular value decomposition in this problem is very large and so instability at small element sizes is likely to result. Additionally, computing the operator with a lower numerical precision (double precision with 64 bits for example) leads to a curve with the same exponential dropoff initially, but with the dropoff halting at a larger size. It is thus likely that these elements are a result only of numerical imprecision.

The exponential dropoff of probability distribution values shown in figure 4.1 has the implication that the Schmidt strength of the rearranged QFT converges to a constant value as the number of qubits in the transform is increased. We compute this strength to be 0.8208. In this measure, the QFT is thus less entangling than a single CNOT or SWAP gate.

The convergence of the bond sizes is shown in figure 4.2 where we plot the mean difference between the size values obtained with a given number of qubits and those of the largest MPO created (44 qubits). It is clear that the values are converging towards the characteristic visible in figure 4.1a. Note that for reasons of speed these results were computed at double precision and so the halting of the convergence at 34 qubits represents the calculation reaching machine precision.

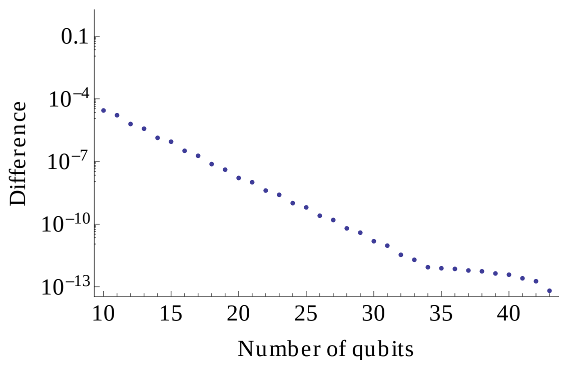


Figure 4.2: The mean difference between the probability distribution of the middle bond of the QFT computed with each number of qubits and that computed with the largest number of qubits (44).

After truncation of the smaller bond elements in the MPOs encoding (4.5), the tensors in the middle of the transform converged to a constant tensor as the number of qubits was increased. This convergence completely specifies tensors in the middle of the transform up to phase rotations which result from the lack of uniqueness of the SVD, which can be easily corrected. The convergence is illustrated in figure 4.3, which shows the mean difference between the absolute values of the elements of the middle tensor of each MPO and the absolute values of the elements of middle tensor of the largest MPO computed (44 qubits). Again, these results were computed at double precision.

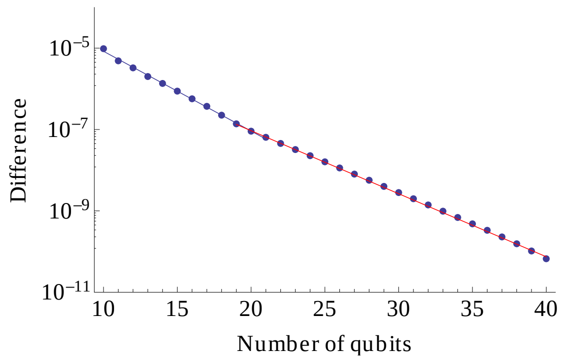


Figure 4.3: The mean difference between the size of the values in a tensor in the middle of a MPO and those of a tensor in the middle of a MPO of the largest size (44 qubits). These differences are normalised by the size of the maximum value in the tensor. Two different decay rates are shown.

Two different exponential decay rates are visible in the plot. The first of these rates is the region at which we must truncate the middle tensor of the largest MPO to compare it to smaller tensors in smaller MPOs while from 20 qubits onwards, the truncation ocurred at a bond size of 30 during calculation of the MPO and so the tensors are the same size.

It is clear from our results that the sizes of each bond in a MPO of the QFT decrease exponentially with increasing bond size. Truncating a bond to size t would thus create an error of $O(e^{-t})$, and $O(n)$ truncations in a n qubit transform would create errors of size $O(ne^{-t})$. To maintain a constant error as the number of qubits is increased, the bond size required would thus be $O(\log(n))$. The convergence towards a common tensor in the middle of the transform implies that it would be possible to find a standard MPO form for the QFT with a relatively small number of qubits determined by a given error tolerance. The middle tensor of this standard QFT could then be replicated a number of times to apply the transformation to any required larger number of qubits.

As noted in section 3.3.1, applying a MPO with a maximal bond size of χ_o and n qubits to a MPS with a maximal bond size of χ_s and n qubits takes time $O(n\chi_o^2\chi_s^2)$. Applying a n qubit QFT to a MPS with maximum Schmidt rank χ, this simulation technique would thus allow simulation of the QFT in $O(n\,(\log(n))^2\,\chi^2)$ time (not including the time required to perform new SVDs which would potentially reduce the bond rank).

4.4 Truncation Errors

Truncation of bonds of even small size will neccesarily introduce error into the representation of an operator. In order to confirm that an efficient MPO simulation of the QFT can be run with the bond size scalings suggested by figure 4.1 without compromising accuracy, it is necessary to quantify this error. It is difficult to quantify the error in a large MPO because the computational cost of calculating any interesting metric will in general grow exponentially with the number of qubits. This is true of any calculation which does not take advantage of the structure of the MPO. For example, many matrix norms require the calculation of the eigenvalues or decompositions of the full matrix of an operator, or maximisation of a function defined on the full matrix. The matrix representation of the QFT is not sparse, and so the exact calculation of such quantities is intractable.

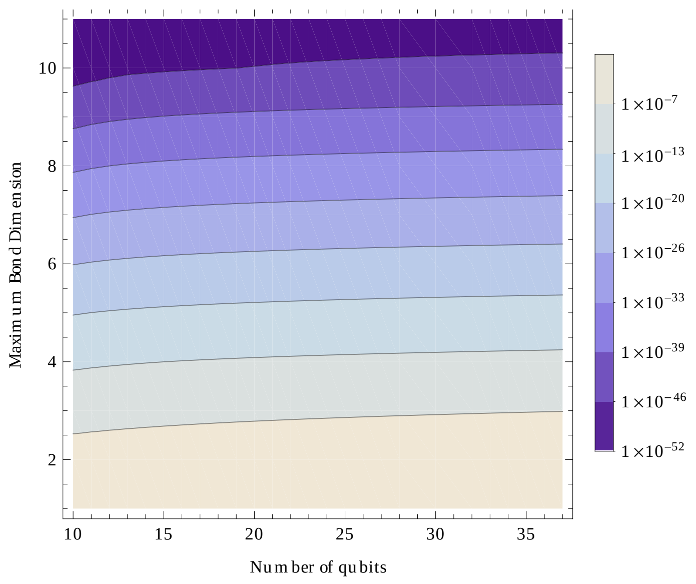


Figure 4.4: The error in the trace inner product between two MPOs of the QFT, one with truncation and one without any truncation.

Instead, we have calculated two less rigorous but more easily computed norms. In order to perform the calculations at large system sizes, these calculations had to be performed with double precision.

Firstly, we computed the Hilbert-Schmidt inner product $\frac{1}{D}\text{tr}(UV^*)$ where U is a MPO representing the QFT and whose bonds are truncated to a given size, V is the same operator but is not truncated and D is the dimension of the Hilbert space. The value obtained measures the inner product between $U|\psi\rangle$ and $V|\psi\rangle$ averaged over all states $|\psi\rangle$. The error in the result $1 - \frac{1}{D}\text{tr}(UV^*)$ is shown in figure 4.4. It can be seen from this calculation that the error drops off exponentially as the bond rank is increased, and increases sub-exponentially as the number of qubits is increased. However, this regime only extends as far as a maximum bond rank of 8, after which the observed error was zero. At these ranks, the average error is thus below the machine precision of around 10^{-16}. It is worth noting that this is only an average measure of error and so does not reflect the worse case error involved in applying a truncated MPO.

The second measurement of error we computed is the amount of error associated with Fourier transforming a periodic state. A periodic state with L qubits and period r takes the form $\sum_{n=0}^{2^L/r-1} |k_0 + nr\rangle$. These states are produced by the modular exponentiation stage of Shor's algorithm. Applying the QFT to a periodic state produces a state which is strongly peaked around the values $\left|i/r\,2^L\right\rangle$ for $i < r$ and so measuring the Fourier Transform of a periodic state reveals the period. This is the basis of Shor's algorithm.

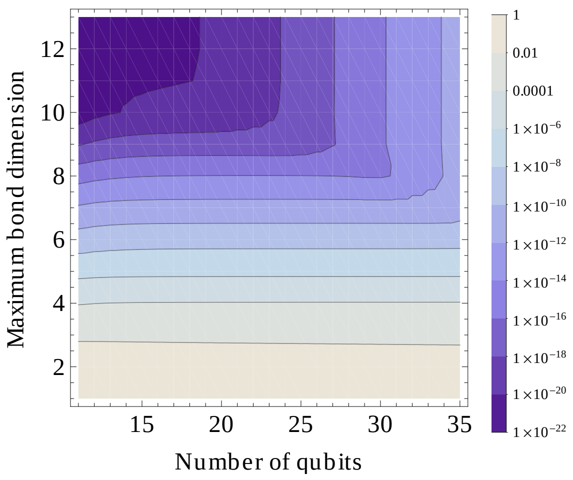


Figure 4.5: The difference between the probability of measuring a peak after simulating a QFT with a MPO on a periodic state and the analytic probability. Shown for a period of 9.

We prepared periodic states with a range of periods and numbers of qubits between 10 and 28 and computed the deviation of the sizes of the peaks from the analytic values. These results are shown in figure 4.5 for 9 qubits. The results were similar for other periods for other values tested ($2 \leq r \leq 15$). As with the trace inner product, the error seems to decrease exponentially as the bond rank is increased at low bond ranks. At higher bond ranks, the error appears to increase quickly as the number of qubits is increased. It is difficult to obtain data with larger numbers of qubits due to the exponential scaling of the calculation of the analytic size of the peaks.

The results in figure 4.1 indicate that many extra bond elements appear at double precision with sizes relative to the largest element of 10^{-13} or less. We would expect that errors observed after simulating the QFT of periodic inputs would be at less than or equal to these levels. This is the case for the range of qubits tested.

4.5 Fragility of the scaling

It is difficult to apply precise phase shifts in a physical quantum computer. As such, we consider the effect of small errors in the controlled phase gates. Doing so also also allows us to determine how robust the scaling of the size of the MPO with increasing numbers of qubits is to a particular kind of noise in the quantum circuit. We prepared MPOs of the QFT, but when applying a controlled phase of θ between qubits a and b, we instead applied a controlled phase of $\theta + \delta r$, where δ is the size of the random error and r is a uniformly distributed random number between 0 and 1. While operational noise in a physical quantum computer is generally modelled by a normally distributed random phase, our choice of a uniform distribution allows us to relate these results with those reported in section 4.3.

We compared the MPOs with errors to those without by fixing the number of qubits and subtracting the singular value vector in the middle bond with errors from the singular value vector of the MPO without errors. Each bond vector with errors deviated from the error-free vector by a roughly constant amount along the length of the vector. As the singular values of the QFT exponentially decay, this leads to characterstics similar to those shown in figure 4.1. We show a set of probability distributions derived from four MPOs with different δ values in figure 4.6.

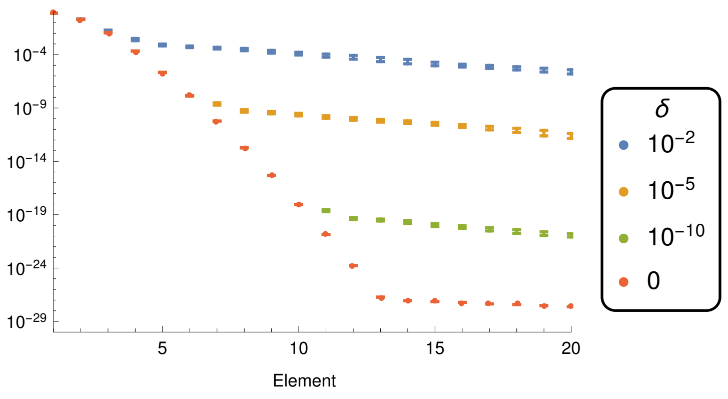


Figure 4.6: The probability distribution derived from the singular values of the middle bonds of MPOs representing the QFT with 15 qubits computed with random phases errors of size δ. These MPOs were produced at double precision.

It is no coincidence that the curves in figures 4.1 and 4.6 look similar. Because controlled phase rotations constitute most of the gates in the QFT (not including swap, which is a permutation of a two qubit tensor), introducing a small random error of order δ to the angle of the phase rotation has a similar effect to limiting the precision of the calculation of the MPO to δ. As such, by varying the value of δ we are doing a similar operation to computing the MPO of the QFT at arbitrary precision.

We show the mean error in the middle bond vector that resulted from varying δ in figure 4.7. These errors were computed by forming 100 MPOs for each value of δ and averaging over the deviations from the error free MPO. Also shown is a linear fit between the logarithm of the error and the logarithm of δ. As this is a log-log plot with a clear linear relationship, the error varies polynomially with δ and the gradient of the fit gives the power of this dependance. In this case the power was found to be 1.031 ± 0.005, an almost linear relationship between the random errors introduced in the controlled phases and the resulting errors in the singular values.

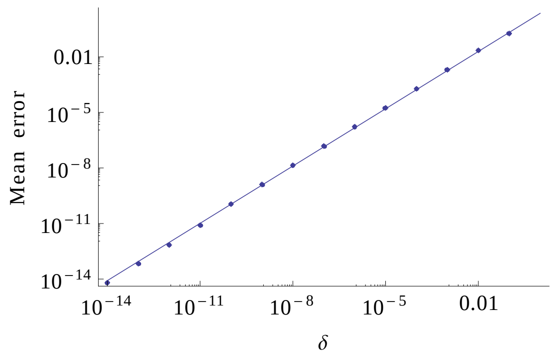


Figure 4.7: The mean differences between the singular value vectors of the middle bonds of the QFT with 15 qubits and random phase errors of size δ, and of the QFT with no phase errors. These values are normalised by the largest singular value in the bond.

The middle bond vectors found in each of the random MPOs generated for each value of δ displayed a similar dependence on element number to those displayed in figure 4.6. Each small additional singular value in the vectors generated with errors will create a random error in the final coefficients of the QFT operator. As the deviations from the more accurate set of singular values result from a random phase, we do not expect that the random errors created by each singular value will add to produce a much larger error. As such, despite there being a large number of additional singular values created by the noise in each case, we expect that if the size of the mean error in the singular value vector is ϵ, the error in the coefficients will be $O(\epsilon)$. As such, a phase imprecision of δ will bring about an overall error in coefficients of the resulting QFT operator of roughly the same size. Conversely, if the coefficients of the QFT are required to a precision of ϵ, the MPO only needs to be computed with numbers of approximately the same precision.

4.6 Reasons for the efficient representation

The fact that ordering the input values of the qubits differently to the output values can lead to a dramatic reduction in MPO complexity raises the question of whether a different ordering to that considered in (4.5) may be optimal. We tested this by constructing MPOs with all possible input qubit orderings for QFTs with up to twelve qubits. In every case the ordering in (4.5) was optimal. For the reasons described above (the output at the nth qubit depends only upon the input at the first n qubits), we expect this to be the case for larger numbers of qubits as well.

It would seem natural to explain the exponential decrease of bond element size shown in figure 4.1 with the small effect of the rotation gates correlating far away qubits. That is, the full QFT introduces correlations across every qubit pairing. However, these correlations take the form of controlled phase rotations and the size of the rotations decreases inverse-exponentially with the one-dimensional distance between qubits. As such, we should be able to neglect long range correlations. We would expect this to cause the sizes of the tensors at the qubits within the MPO to be almost entirely unaffected by the number of far-away qubits. This is the idea behind the AQFT [100], where the number of controlled phase gates conditioned upon each qubit, henceforth the bandwidth, is set at a fixed value irrespective of the number of qubits in the transform.

However, we constructed MPOs using a nearest neighbour quantum circuit of the AQFT and found that the bond ranks produced were larger than those produced for the full transform. The maximum bond ranks for a series of AQFTs after truncation are shown in figure 4.8. Maximum bond ranks in an AQFT increased by a factor of 2 per additional controlled phase rotation included. This increase levelled off in the middle of the transform but still quickly became computationally intractable. Furthermore, the trace inner

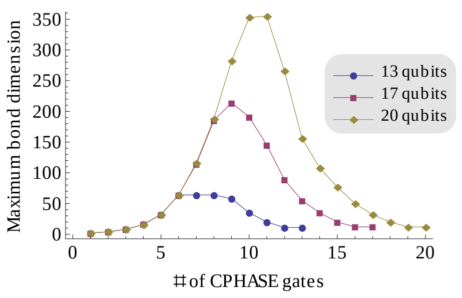


Figure 4.8: The maximum bond rank in MPOs corresponding to AQFTs with different numbers of qubits. Each AQFT was constructed with a different maximum number of controlled phase gates conditioned on each qubit.

product between an operator truncated at any bond rank and a series of operators with reduced bandwidths was a maximum for the full transform and decreased monotonically as the bandwidth decreased. It thus seems that the low required bond ranks observed in (4.5) are a feature of the full QFT.

While the low required bond rank of the QFT cannot be attributed entirely to decreasing phase rotations, the size of these rotations and the rate of their decrease are important. We found that transforms with the same structure as the QFT but with phase rotations decreasing as $\exp\left(2\pi i/k^n\right)$ instead of $\exp\left(2\pi i/2^k\right)$, where k is the qubit distance and n an integer, did not display the characteristic dropoff of bond size. Rather, the required bond ranks appeared to increase with increasing numbers of qubits, presumably until the phase rotations become smaller than machine precision. Rotations of the form $\exp\left(2\pi i/n^k\right)$ for $n \geq 2$ still lead to Fourier transforms, although not over $\mathbb{Z}_{2^m}$, and were found to still lead to an exponential dropoff in bond element size. The rate of this dropoff increased as n increased.

As such, while the small bond rank required to accurately represent the QFT with a MPO is not due solely to the decreasing size of the phase rotations used, it is related to them. It is likely the exponential dropoff of bond size is the result of a symmetry in the structure of the QFT. In order to obtain a low bond rank it is necessary to have phase rotations which decrease at least exponentially with the distance between qubits and to have the same phase rotation for each conditioned gate at a given linear qubit distance.

4.7 Implications and complexity

While we have not proven that the QFT can be efficiently represented as a MPO, our numerical results are strongly suggestive of this. It appears that the numerical error associated with the very small amount of truncation required for a tractable representation is very close to zero over a range of numbers of qubits. Additionally, the differences between a matrix in the middle of each operator and an adjacent matrix decreases as the system size increases.

Together, these results suggest that a MPO representing a QFT for an arbitrary number of qubits can be created from the MPO representation of a QFT of a smaller size. With an appropriate bond rank, this would allow the QFT to be performed on a MPO with maximum Schmidt rank χ with computational cost $O(n\,(\log(n))^2\,\chi^2)$, as noted in section 4.3. It could similarly be performed on weakly correlated mixed states. Our method allows the QFT to be efficiently simulated in a straightforward fashion in any case in which the qubits are ordered linearly.

Application of the QFT to a MPS of n qubits with this method increases the bond rank by at most a factor scaling as $O(\log(n))$. Denoting this factor by d, the application of m QFTs increases the bond ranks by a maximum factor of d^m. As such, the application of a constant number of QFTs can be efficiently simulated with a large number of qubits. These QFTs can be interspersed by quantum circuits that do not increase the Schmidt rank.

Our results strengthen earlier work. In [127] the AQFT is show to be classically simulatable in polynomial time, although an explicit scaling is not derived. Our method of simulation uses the full QFT and has a more advantageous scaling with respect to the number of qubits of $O(n\,(\log(n))^2)$.

In [127] a condition is also derived for when two efficiently simulatable quantum circuits composed may be efficiently simulated. From this condition it follows that any circuit composed of a constant number of AQFTs and log-depth limited interaction range circuits can be efficiently classically simulated. We provide a different perspective on the composability criteria. That is, our method makes explicit the scaling of the cost of the QFT with the Schmidt rank of the bipartitions in the input state. We have shown the difficulty of simulating the QFT to be mostly determined by the complexity of the state being transformed. A log-depth limited interaction circuit will produce an input state with small Schmidt ranks across each bond partition, and so the previous result follows from our results.

That the QFT can be represented to very high fidelity with a MPO with limited bond ranks implies that the QFT can produce only a limited amount of entanglement. This conclusion was originally shown in [42], however our methods are more straightforward.

We take a different approach to simulating the QFT to that presented in [71, 72], which uses an algebraic extension of stabilizer techniques to efficiently simulate a range of circuits including a polynomial number of QFTs. These results also allow the QFT to be simulated on highly entangled states and on different abelian groups. Our results take a tensor network approach and only allow a constant number of QFTs to be simulated efficiently on states with entanglement that grows polynomially with the number of qubits. Additionally, our results allow a classification of the difficulty of simulating the QFT on arbitrary input states to be calculated.

With respect to the question of where the quantum speedup in Shor's algorithm originates, our results provide further evidence that it originates in the highly entangled state generated by modular exponentiation. As will be discussed in the next section, periodic states are generated by modular exponentiation, and a state of period r will have bond ranks in a MPS of r. As the maximum period of a modular exponentiation process factoring a number N scales as $O(N)$ [130], states with very high Schmidt numbers are generated. These states are very difficult to represent in a MPS and thus are very difficult to Fourier transform. This conclusion is similar to that reached in other works such as [127, 132]. It is additionally worth noting that while our method makes very clear the connection between the Schmidt rank of the input state and the difficulty in Fourier transforming it, the same conclusion can be drawn about the computational speedup from the results of [127].

Chapter 5

Simulation of Shor's algorithm

In this chapter we will simulate Shor's algorithm in the presence of Z errors using MPS representations. In doing this, we will build upon a previous work in which the MPS simulation of Shor's algorithm was introduced. We will increase the speed of our simulation by commuting Z errors to a limited number of points in the algorithm and by using the results of chapter 4.

In section 5.1 we will describe previous work in which Shor's algorithm has been simulated and the effects of errors observed. We detail our simulation approach in section 5.2 for pure state simulations of Shor's algorithm, and in section 5.3 for simulations in the presence of noise. In section 5.4 we perform noisy simulations of Shor's algorithm and present the results with respect to the success of the algorithm and in section 5.5 we quantify the entanglement present in our MPS simulations.

5.1 Previous Results

Shor's algorithm is the prototypical example of a non-trivial quantum algorithm which offers an exponential time saving over the fastest known classical alternative. As such, the algorithm has been studied in many different ways. We are principally concerned with the simulation of Shor's algorithm and with the conclusions which can be drawn from these simulations. Shor's algorithm has been targeted by increasingly large parallel simulations [134, 135, 136, 137, 138, 139]. Of these, the most advanced is given by [138, 139] in which a 128 core cluster computer is used to simulate Shor's algorithm for N up to 1034273 (this simulation uses 40 qubits) by performing modular exponentiation classically. The authors also simulate all of Shor's algorithm expanded with the Beauregard circuit for N up to 57. A larger simulation is reported in [136], in which up to 4096 processes and 1TB of memory is used to simulate quantum computers containing up to 36 qubits. Another promising approach to simulating Shor's algorithm is presented by Wang, Hill and Hollenberg [140] (henceforth WHH), who uses MPSs to simulate Shor's algorithm for up to 36 qubits in a single process with 2GB of memory. In this chapter we will follow the approach of WHH, extend their method to include the simulation of a particular error model, and report some preliminary results.

5.1.1 MPS simulation

While most simulation of Shor's algorithm takes place with a state vector approach, a MPS simulation is detailed in the work of WHH. This simulation is found to be much easier to perform than a state vector simulation. However, it is only possible to simulate pure state evolution using this approach. In this chapter we will present a generalisation of the MPS simulation method to allow simulation of dephasing noise, and present further results indicating the difficulty of incorporating other kinds of noise. To provide a base for our work, we will summarise the method and results of WHH here.

In WHH, a MPS simulation is conducted of Shor's algorithm in circuit form. The state is first prepared with $2l$ qubits in the upper register and a qudit in the lower register of dimension 2^l. The lower qudit is kept in sparse form, so that only populated computational basis states are recorded. As modular exponentiation results in the population of r lower register states, where r is the period of the process, this results in the lower qudit only ever reaching a dimension of r.

Modular exponentiation is applied by contracting the rightmost qubit in the upper register with the lower register qudit, applying the modular exponentiation matrix and then performing a trivial decomposition between the qubit and qudit:

$$A_{m\times n} = \begin{cases} A_{m\times n} I_{n\times n}, & m \geq n \\ I_{m\times m} A_{m\times n}, & m < n \end{cases} \tag{5.1}$$

The control qudit that was used is then moved to the left hand size of the MPS by a series of SWAP gates and QR decompositions. If k states are populated in the lower register after a modular exponentiation step, this results in the MPS having bond dimension k between the upper and lower register, and on every bond between the lower register and the final position of the control qubit after it has be swapped to its final location. By changing the order in which modular exponentiation steps are performed, time savings are made by minimizing the value of k for as long as possible, although it must grow to r by the end of the modular exponentiation (this is found to be the minimal bank rank possible at this point in the algorithm). After modular exponentiation, the lower register is measured and the sizes of the bonds in the upper register minimised by a series of QR decompositions.

The QFT is applied by using the nearest neighbour QFT circuit (shown in figure 2.12) and applying it to the state which results from modular exponentiation. The author assumes that the MPS after the QFT will be maximally entangled, and so uses trivial decompositions (5.1) after each combined controlled phase and swap gate step (our results in chapter 4 indicate that this is not correct). The MPS is finally contracted fully to write the state as a single amplitude vector so that the amplitude of the computational basis state can be assessed. Simulation of Shor's algorithm with a MPS is concluded to require $O(2lr)$ space before the Fourier transform and $O(2l \cdot 2r^2, 2^{2l})$ space in total, a saving on the $O(2^{3l})$ required for a state vector simulation. A time scaling is not derived.

5.1.2 Entanglement

There have been many papers discussing the contentious issue of whether there is a connection, and what form it takes, between entanglement and the computational speedup of quantum algorithms over classical algorithms. However, there have been relatively few papers explicitly quantifying the amount of entanglement in Shor's algorithm. Firstly, Parker and Plenio [141] show that a version of Shor's algorithm with a single pure qubit and the others in a mixed state is viable. This simulate this version of the algorithm and show that the entanglement is still non-zero.

Kendon and Munro [142] simulate instances of Shor's algorithm with values of N between 15 and 119. For $N = 15$ and $N = 21$ they sample bipartitions to determine the negativity of each bipartition in the upper and lower registers with given numbers of qubits in the two halves. They also calculate the entanglement of formation for each pair of qubits. For larger N values they calculate the negativity of bipartitions of 1 qubit against the others in the upper register before and after the QFT, and quantify the decrease in the entanglement.

Shimoni et al [143] and Most et al [144] consider the groverian measure of entanglement applied to Shor's algorithm. This entanglement measure expresses the distance between a state and the nearest product state. They perform simulations of Shor's algorithm with N between 3 and 200 and determine that the entanglement is always less than $\sqrt{1-1/(2N)}$ and that the entanglement remains close to constant during the QFT stage of the algorithm. The latter paper develops theoretical approximations for the entanglement of periodic states and explains why the QFT does not increase the entanglement.

5.1.3 Shor's algorithm in the presence of noise

One of the applications of simulating Shor's algorithm is to study the affect of noise or imperfections on the operation of the algorithm. Research in this area has occurred for some time, and most authors use a state vector approach to simulate the algorithm in the presence of their chosen error model. The earliest of these works are [145], which simulates dissipative noise for up to 15 qubits; and [146], in which phase drift errors are simulated in a ion trap quantum computer. A form of Shor's algorithm with with one pure control qubit and a series of mixed qubits is considered and simulated in [147], in which the amount of entanglement is quantified with increased mixing. The work of Wei [148] simulates dynamical evolution during a time delay between gates for $N = 4, 15, 21$ and 31. Devitt et al [149] examine depolarizing noise in Shor's algorithm for between 14 and 20 qubits and determine the maximum permissible number of errors numerically. Garcia-Mata et al [150, 151] consider static imperfections and couplings between qubits and find that the algorithm is viable up to a strength of coupling ϵ which decreases polynomially with the number of qubits. The latter of these two simulations uses only a single control qubit and so is able to perform simulations with up to $N = 205193$ (a simulation involving 19 qubits). Another class of work related to the affect of noise and imprecision in Shor's algorithm are those which consider the affect of using the AQFT instead of the full QFT [129, 152, 130]. This issue is often approached from the point of view of reducing the number of gates in a quantum computer [138], but can also be phrased with relation to imperfect gate operation.

5.2 Optimised MPS simulation

We perform error-free MPS simulation with a similar method to that of WHH. The upper register is divided into $2l$ tensors, one for each qubit. The lower register is considered as a single 2^l dimensional qudit, however the amount of information stored is minimised by only tracking the non-zero coefficients in this register. The lower register is always kept at the rightmost side of the MPS as it has the highest quantum dimension, and so we can limit the size of the overall MPS by only allowing the bottom register to have one bond. We split Shor's algorithm into two stages, the modular exponentiation and the QFT.

5.2.1 Modular exponentiation

We consider modular exponentiation as a series of gates between the rightmost qubit in the upper register and the lower register. The state of the system after the modular exponentiation step is:

$$|\psi\rangle = \frac{1}{2^l} \sum_{j=0}^{2^{2l}} |j\rangle \big|x^j \bmod N\big\rangle. \tag{5.2}$$

There are r distinct values of $x^j \bmod N$ and so if we denote $f(j) = x^j \bmod N$ and the r distinct values of $f(j)$ by $f_1 \dots f_j$, we can rewrite (5.2) as

$$|\psi\rangle = \frac{1}{2^l} \sum_{i=1}^{r} \left(\sum_{j:f(j)=f_i} |j\rangle \right) \otimes |f_i\rangle. \tag{5.3}$$

Each j maps to only one value $f(j)$ and so each of the states in the left and right parts of (5.3) are orthogonal, and it is clear that the Schmidt rank of $|\psi\rangle$ across this bipartition is r. As r is the number of non-zero states in the lower register, we can obtain a MPS with minimum bond dimensions across the bond between the upper and lower registers by performing the trivial decomposition (5.1). Our simulations indicate that the Schmidt rank of the upper:lower register bipartition of the system is also the number of states with non-zero coefficients in the lower register at other times during the algorithm, and so a trivial decomposition will be appropriate at all times when performing modular exponentiation gates between the upper and lower registers.

Starting from the least significant bit (we do not apply WHH's reordering of the modular exponentiation gates), we apply modular exponentiation gates between the registers. After each modular exponentiation step, we move the control qubit used to its final position, the space directly below the previous control qubit used. It is possible to perform this action with a series of SWAP gates and SVD decompositions, however this is computationally inefficient. After k modular exponentiation gates have been applied, the k leftmost qubits of the upper register will have been used as control qubits and so will be entangled with the lower register. The remaining $2l - k$ qubits between the bottom of the upper register and the previous control qubit will not have interacted with the system and will be in the state $1/2\left(|0\rangle + |1\rangle\right)$. The Schmidt rank between the k qubits that have interacted and the lower register will be equal to the number of non-zero values in the lower register r_k and so this will be the size of each bond between the k top qubits in the upper register and the lower register. These bonds join qubits that have not interacted, so we set the tensor of each of the qubits $\{k \ldots 2l\}$ to be $\Gamma^{[a]\alpha_{a_1}\alpha_a}_{i_a} = 1/\sqrt{2}\,\delta_{\alpha_{a-1}\alpha_a}$. We can write the vector on each bond as $\lambda^{[a]} = (1, \ldots, 1)$. Using this qubit reordering technique, our overall procedure to perform the kth modular exponentiation gate can enumerated:

1. Contract the control qubit, the bottom qubit of the upper register, with the sparse qudit of the lower register. As the control qubit has not yet interacted with any other part of the system, it will be in the state $1/2\left(|0\rangle + |1\rangle\right)$.

2. Apply the modular exponentiation gate. This step can be performed either by forming a matrix for the gate and applying it by multiplication, or by moving regions of memory so that in the jth gate, $|0\rangle|i\rangle \rightarrow |0\rangle|i\rangle$ and $|1\rangle|i\rangle \rightarrow |1\rangle\left|i \times a^{2^j} \mod N\right\rangle$. Moving memory exploits our knowledge of the action of the modular exponentiation operator and is faster than explicity forming a gate and multiplying it by the combined two qubit tensor.

3. Decompose the new combined two qubit tensor into two single qubit tensors. As the qudit representing the bottom register is sparse, we will have the values in which it has a non-zero coefficient ordered as $\{r_i\}$. We can then perform a trivial decomposition into the left tensor, representing the control qubit; and the right tensor, representing the bottom register. If we denote the combined two qubit tensor after gate application as $\Gamma^{[lr]\alpha}_{i_l i_r}$ where α is the bond on the left of the two qubit subsystem, and i_l and i_r are the states of the left and right constituent systems, we can write the decomposition as:

$$\Gamma^{[l]\alpha\beta}_{i_l} = \Gamma^{[lr]\alpha}_{i_l\beta}, \quad \Gamma^{[r]\beta}_{i_r} = \delta_{\beta i_r}, \quad \lambda_\beta = 1. \tag{5.4}$$

4. Move the control qubit to location k, the location immediately to the right of the previous control qubit. Copy the new tensor $\Gamma^{[l]\alpha\beta}_{i_l}$ to the kth tensor location in the MPS and move every qubit in the range $[k, 2l-1]$ (note that $2l$ was the position occupied by the control qubit) to the right by one site. If the number of values in the lower register r_k, which is also the bond size of the new bond β, has changed from the previous value r_{k-1}, update every tensor while moving them one site to the right. The elements of each new tensor will be $\Gamma^{[a]\alpha_{a-1}\alpha_a}_{i_a} = 1/\sqrt{2}\,\delta_{\alpha_{a-1}\alpha_a}$, where the bonds α_{a_1}, α_a will now have size r_k.

We note that in our procedure it is not strictly necessary to have $2l$ qubits in the upper register at all times. As each qubit is in the state $H|0\rangle$ until it is used as a control qubit and interacts with the bottom register, it is only necessary to track qubits while they are interacting with the lower register and after the interaction. It would be possible to prepare a state of one qubit and the lower register and interact the two, and then to add a qubit between them. This qubit would interact with the lower register and another qubit would be added. This procedure would continue until there had been $2l$ interactions.

We also note that we have applied modular exponentiation gates in large single units. This is not possible on a physical quantum computer, where modular exponentiation needs to be decomposed into fundamental gates to be performed. There are several different ways to perform this decomposition on each quantum architecture, for example linear nearest neighbour quantum computers and a 2D array of qubits. Each decomposition requires a different numbers of gates and a different number of ancillary qubits.

We could perform simulations of linear nearest neighbour gate decompositions of modular exponentiation procedures with MPSs, but we have chosen to perform a higher level simulation. A simulation of a particular decomposition would be computationally much more difficult than a simulation such as ours, limiting the problem size that could be studied. Additionally, while not translating directly to gates that could be performed on a quantum computer, a higher level simulation is interesting to assess the strengths of simulation methods and to perform simulations of algorithms with very low levels of noise. We will perform such a simulation in this chapter.

Time required

To analyse the time complexity of simulating modular exponentiation, we will consider separately the time complexity of initial preparation of the state, and each of the steps of performing a modular exponentiation gate. Preparing the state initially requires each of the $2l$ qubits and the qudit to be prepared in a product state and so will take time $O(l)$. To perform the kth modular exponentiation step, we have to first contract the control qubit with the lower register. With dense tensors this would take $O(r_{k_1}^3)$ time, but as the control qubit is in a product state and the lower register tensor is sparse, this contraction will only take $O(r_{k_1}^2)$ time, and will yield a combined tensor of $O(r_{k_1}^2)$ size. We can accomplish the modular exponentiation itself by moving areas of memory around, and so this will take $O(r_{k_1}^2)$ time, the same scaling as the size of the matrix. The final decomposition requires a relabelling of the new tensor and the creation of a new identity tensor for the lower register, which will take time $O(r_{k-1}r_k)$ and $O(r_k^2)$ respectively. We may then have to revisit and reform each of the unused tensors, which will take $O(nr_k^2)$ time, however we will not include this in our overall scaling as this step does not have to be performed many times and so the computation time may be amortised over the simulation.

Each of the steps involved in modular exponentiation thus involves a time scaling of $O(r_a^2)$ for some a with $r_a \leq r$. As the value r_k will quickly approach the maximum r and there are $2l$ modular exponentiation gates in total, the preparation of the state and the simulation of the modular exponentiation will take a total time of $O(lr^2)$. The MPS produced at the end of the modular exponentiation will have a space complexity of $O(lr^2)$.

5.2.2 QFT

After the simulation of the modular exponentiation, we must perform a QFT. Shor's algorithm allows the state of the lower register to be measured after the modular exponentiation but this is not necessary for the success of the algorithm. We measure the state of the lower register by selecting a value out of the r non-zero values we keep track of. In a pure state simulation of Shor's algorithm, each of the lower register values is equally likely, so we are free to select one at random. Because the bond between the upper and lower registers is formed with a trivial decomposition, measuring the mth value on the lower register will cause the last tensor to be removed, and the mth value of the last bond to be selected (all other values in this bond are set to zero). A series of SVDs moving from right to left and then left to right will return the MPS to canonical form. The bond sizes in the canonical MPS must be smaller than in the MPS before measurement because each quantum system only has a dimension of 2, while all of the bonds including those on the rightmost part of the upper register were of size r before the measurement. As such, measuring the lower register reduces the size of the MPS representation of the system.

Instead of performing the QFT on our system by applying quantum gates to the MPS, we use the MPO form of the QFT. In chapter 4 we found that the QFT can be simulated on a state of n qubits with Schmidt rank χ in $O(n\chi^2(\log(n))^2)$ time. Here we have a state with a Schmidt rank less than or equal to the period of the modular exponentiation r, and so the QFT can be simulated in $O(nr^2(\log(n))^2)$ time by applying the MPO directly to the MPS. The MPO for the QFT can be calculated ahead of time and then applied in a large number of different states, so we do not include the cost of calculating it in the time required to perform our simulation.

After applying the QFT, we are free to perform further computation or quantum gates to the state, or to perform contractions to determine any desired coefficients. We found in chapter 4 that if only a single coefficient is required, the MPO of the QFT takes the form of a bond size one projector projeting to a particular quantum state. We can calculate the inner product between this quantum state and our MPS by contraction to determine the coefficient of the desired index after the QFT.

Time required

As indicated above, to apply the QFT MPO to the MPS resulting from the modular exponentiation will take $O(nr^2\,(\log n)^2)$ time, after which additional computation can be performed or a series of contractions performed. If a limited number m of coefficients is required (for example to determine the probability of obtaining useful output for Shor's algorithm but not what the other coefficients are), we can form a series of bond size one MPSs and take an inner product with the state after modular exponentiation for each coefficient. As these projectors can be prepared earlier and used an unlimited number of times, the cost of their preparation can be amortized across a large number of Shor's algorithm simulations. The process of obtaining a coefficient after modular exponentiation thus takes the same amount of time as a contraction of a single coefficient from the MPS. This takes time $O(lr^3)$, and so obtaining m coefficients will take time $O(mlr^3)$

5.3 MPS Simulation with errors

Shor's algorithm is relatively easy to simulate with a MPS primarily because we are free to select the parameters of the algorithm (the value of N to be factored and the additional number a) to ensure that the order of the modular exponentiation r is small. In doing so we ensure that limited entanglement develops. The presence of general noise modelled by density matrix evolution or averaging over many pure state simulations of the noisy algorithm will decrease the amount of entanglement. However, any individual pure state simulation of Shor's algorithm with a stochastic noise model will have a dramatically increased Schmidt rank to the noiseless case. This problem is worse the larger that r is, but is still present in cases with small r values. This would seem to present a paradox in which adding noise to Shor's algorithm dramatically increases entanglement. This is not the case, because each individual stochastic pure state simulation is physically meaningless. It is only by averaging over a statistically large sample of random noisy pure state simulations that a physically meaningful result can be calculated.

5.3.1 Z errors

We will first consider the effect of dephasing noise on the MPS simulation of Shor's algorithm. General dephasing noise can be introduced by assigning a probability p_z that a Z gate can occur after any quantum gate, or on any qubit every time-step during general evolution. To model this, we thus must allow a Z gate to be placed between any two quantum gates in our circuit. Shor's algorithm is composed of the modular step and the QFT. Modular exponentiation in turn consists of Hadamard gates, controlled modular exponentiation gates and SWAP gates. A Z gate will commute through modular exponentiation gates, and so we are free to change the order of these two gates in a quantum circuit. A SWAP gate will change the qubit that the Z gate is applied on, but not any other part of its effect. As such, we can change the order of a Z gate and a SWAP gate as long as we change which qubit the Z gate is applied on. A Z gate does not commute with a Hadamard gate, and so we may not change the order of these two gates in a quantum circuit. Given a circuit composed entirely of SWAP gates and modular exponentiation, we can thus commute a Z gate from wherever it is placed in the circuit to the outside of the circuit (the start or finish depending upon which we prefer) as long as we keep track of which qubit we are applying the phase to when we encounter swap gates. If we consider the input to Shor's algorithm to be the state $H^{\otimes n}|0\rangle \otimes |1\rangle$, then we can move any Z gates that occur to the end of the modular exponentiation step

As such, to simulate modular exponentiation in the presence of only dephasing noise, we can create the MPS and perform modular exponentiation once, and then copy the MPS multiple times and apply the errors to the different copies. The error probabilities can be straightforwardly calculated based on the per time step or per gate error probabilities, the number of gates that has acted on each qubit, and the number of time steps that have elapsed.

The QFT is composed of Hadamard gates, controlled phase gates and SWAP gates. Z gates commute with controlled phase gates, and as above they can be commuted through a SWAP gate if we change which qubit the Z gate is being applied to. However, they do not commute through Hadamard gates, and so the order of a HZ or ZH series of gates cannot be changed without introducing other gates. In the canonical decomposition of the QFT, each qubit undergoes one Hadamard, after which a conditional

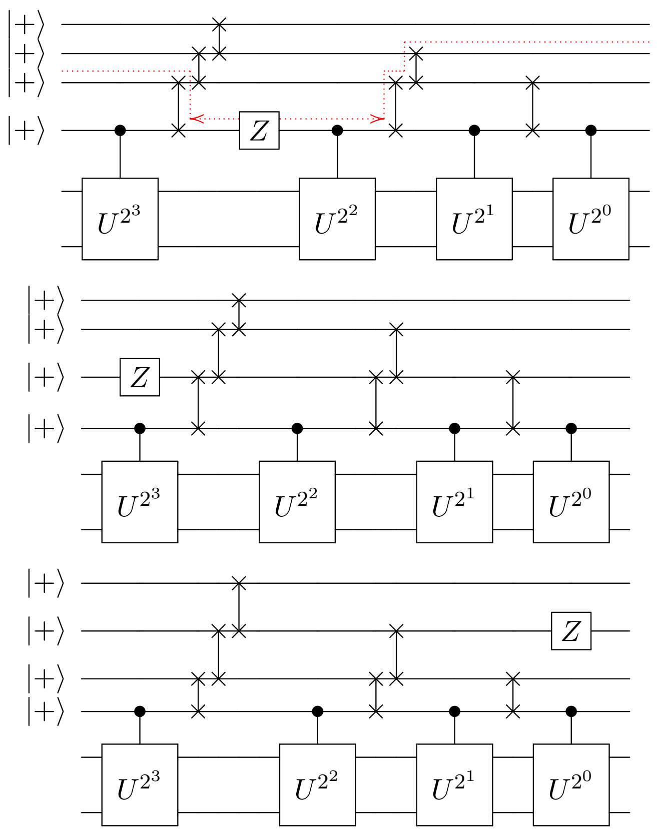


Figure 5.1: (a) A nearest neighbour modular exponentiation circuit with a single Z error, and a line showing the path the Z gate can take to move around the circuit due to commutation relationships (the dotted red line). Also shown are two equivalent circuits with the Z operator moved to the beginning (b), or end (c), of the circuit.

phase is applied between it and each qubit lower in the one dimensional ordering. Considered this way, we can consider Z errors that occur in two groups: those that occur before the Hadamard on their qubit, and those that occur afterwards. If a Z error occurs before the Hadamard, it can be commuted backwards through all of the controlled phase interactions, and so it is equivalent to a Z error occurring before the QFT has begun. Similarly, if a Z error occurs after the Hadamard, it can be commuted forwards through the controlled phase interactions, and so it is equivalent to a Z error occurring after the QFT has finished.

Expressing the canonical decomposition of the QFT in a LNN architecture changes the way that the circuit appears by introducing SWAP gates. However, because a SWAP gate changes the qubit on which we consider Z

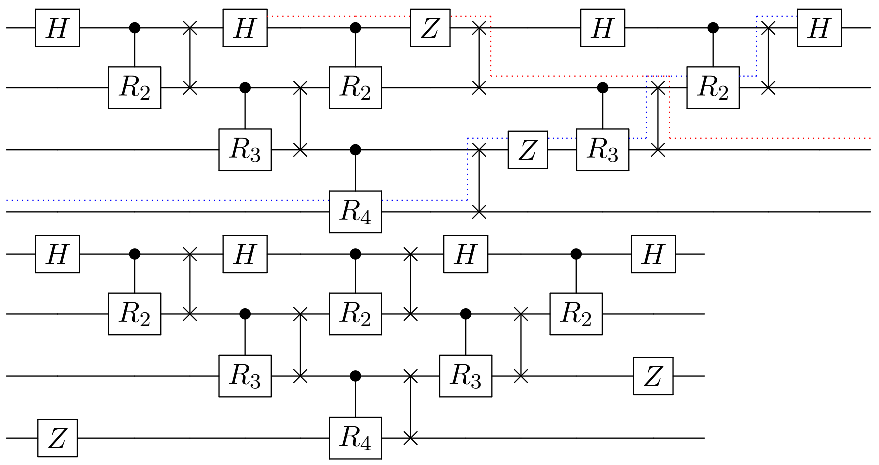


Figure 5.2: The LNN QFT circuit for four qubits with two Z errors. (a) Red and blue lines indicate the locations these errors can be commuted to and in (b) the errors have been moved to the edges of the circuit.

errors to have occurred, the action of Z errors is essentially the same as on the circuit in which arbitrary qubits interact. That is, we can still move all of the Z errors to locations before or after QFT. We illustrate this in figure 5.2, which shows a linear nearest neighbour circuit for the four qubit QFT with two Z errors. The dotted blue and red lines indicate locations to which each error can be moved to give a circuit encoding the same operator. For each error, moving the Z gate in one direction will eventually result in being able to move no further due to the presence of a Hadamard gate, while the other direction will reach the beginning or end of the circuit. By classifying errors into occurring before or after the Hadamard gate on their qubit we can thus commute them to the beginning or end of the QFT circuit.

Given that we can commute Z errors out of the QFT circuit, we can follow a similar simulation strategy to simulate the overall noisy pure state evolution to that employed with modular exponentiation. To elucidate our technique, we consider simulation by the graph contraction paradigm [41]. We can consider the entire quantum circuit as a graph where input states, output states and gates are nodes, and the edges connect interacting gates on each qubit to form the quantum circuit. Each node is associated with a vector (input and output states) or a matrix (gates). In this structure, obtaining a coefficient for a set of indices at the end of the simulation corresponds to setting the output state vectors to be projectors and then contracting the graph by matrix and tensor multiplications.

A MPS simulation is a slight modification of this paradigm in which one gate at a time in contracted into the input nodes, and then decompositions are performed to preserve the graph structure of the input nodes. However, the simulation will return the same result regardless of the order in which we contract the nodes of the graph. As such, we are free to contract any nodes together at any time in the simulation, as well as to perform decompositions on any node which will split it into a series of smaller nodes. In saying this, we are explicitly considering a quantum circuit as a tensor network. A simulation may be optimised by contracting and decomposing nodes in the most beneficial way.

If we need to perform several simulations on closely related circuits (which we consider as tensor networks), we may contract those parts of the circuit which are the same for all simulations, and then only perform the contractions that are different for each separate simulation. In the case of Z errors in the QFT, we thus contract the modular exponentiation into the input state MPS, and the QFT into a MPO. For each particular set of errors, we apply those after the modular exponentiation and before the QFT to the modular exponentiation output MPS, and then apply the QFT MPO. Further Z errors can be applied after this, and then further computations to the output state. If we only require a small set of coefficients, we can produce projector MPOs of these coefficients from the QFT MPO, and then take an inner product with the modular exponentiation output MPS after the Z errors. The final set of errors can be applied by multiplying by -1 any coefficient that has a 1 as its jth index if a Z error occurs on the jth qubit after the QFT.

5.3.2 X errors

Similarly to dephasing noise, we can introduce a second kind of noise to our MPS simulations by assigning a probability p_x that a X gate can occur after any quantum gate, or on any qubit during every time-step of simulation. However, we cannot apply the same techniques which made Z errors tractable to simulate in large instances if we wish to simulate X errors. While the Z gate commutes with the constituent gates of modular exponentiation and most of those of the QFT, X does not commute with most gates of either of these. More specifically, the X gate does not commute with the control of modular exponentiation gates or with the controlled phase rotation gates and Hadamards used in the QFT. We would thus need to incorporate X errors in the middle of the simulation of the quantum circuits performing modular exponentiation and the QFT.

The difficulty of simulating Shor's algorithm in the presence of X errors depends strongly on the qubit and time-step at which the error occurs. As such, we will describe several different kinds of errors that have different effects on the difficulty of performing the simulation.

We will first consider X errors occurring during the modular exponentiation. Applying a X gate to any qubit in the top register at any time-step during modular exponentiation will have no affect on the computation. This is because at all times during the exponentiation, the top register is in the state $|+\rangle$, which is stabilised by X. Applying a X gate to any qubit in the bottom register will in general increase the number of populated elements in the bottom qudit. As the Schmidt rank of the system is essentially the number of non-zero elements in the bottom register, this will have the effect of increasing the Schmidt rank, and so the difficulty of performing the simulation.

To see why an X error in the bottom register increases the number of non-zero values in the bottom register, we consider the action of modular exponentiation. If the period of the modular exponentiation process is r then the bottom register will have non-zero coefficients for r different values. In this context, each modular exponentiation gate is a cyclic permutation between these r values. We make simulation of Shor's algorithm tractable by selecting values of a for each N which generate low periods. However, if a X gate is applied to the bottom register during the modular exponentiation, it will transform each element of the cycle $\{i_1, \ldots, i_r\}$ into a different number. As we perform simulations with low values of r, these transformed bottom register values will not in general be elements of the original cycle. Indeed, for a bottom register with a large dimension, the transformed values may be elements of different cycles. Later modular exponentiation gates populate and cyclically permute each of these new cycles, and after few X the bottom register can become fully populated. This causes the Schmidt rank of the state of the bipartition between the upper and lower register to become maximal, and so the simulation is no longer tractable for large N. From performing simulations with X errors occurring in the bottom register, we find that each X error roughly doubles the number of populated values in the bottom register, and so doubles the bond rank between the two registers. As such, it is not tractable with a simple MPS method to simulate Shor's algorithm in the presence of X errors in the bottom register. It is possible that with modifications such a simulation could be performed.

We now consider the affect of X errors on the simulation of the QFT. It is not possible to commute X errors in the QFT backwards or forwards through any gates beside SWAP gates. That is, the X gate has a non-zero commutator with both the control and target part of a controlled phase gate, as well as with Hadamard gates. To perform a simulation of the QFT including X gates we thus have two possible approaches:

- To simulate the QFT gate by gate including the required X gate. To perform this simulation more quickly for many qubits we may break the QFT up into stages and calculate the MPO of each stage. The MPOs may be applied to each noisy state interspersed with X errors. In general the presence of errors will increase the bond dimension of the state. However, we have not performed detailed simulations to determine precisely how the dimension is changed.

- To calculate the MPO of the QFT including a X error. This MPO may either be calculated by multiplying MPOs encoding smaller stages of the MPO interspersed by X errors, or may performing a full gate by gate multiplication on an identity MPO. Our results in chapter 4 indicate that the efficient MPO representation of the QFT can be compromised by the presence of a small amount of noise. We would thus expect that the MPO representation of the QFT with any X errors would not be an efficient representation. As such, we would expect this to be an inefficient simulation method.

We note that it would be highly unusual for X errors to be present during the QFT but not during the modular exponentiation. We have shown above that MPS simulations of modular exponentiation including X errors are intractable for large N. As such, whether or not QFT simulations involving X errors are tractable does not affect the overall tractability of a large simulation of Shor's algorithm involving X errors.

5.4 Success probability with errors

We perform simulations of large instances or Shor's algorithm in the presence of Z errors. This error model can be physically motivated by noting that the physical processes leading to Z generally occur on a timescale which is much shorter than that leading to X errors. As such, Z errors are much more likely to occur than X errors and the success of quantum computers operating in this regime which are not fault tolerant depend upon the impact of Z errors on the success of algorithms. The use of MPSs allows us to easily simulate Shor's algorithm in the presence of Z errors, due to the properties described above.

We simulate Shor's algorithm with a fixed probability of a Z error occurring at each time-step. We perform these simulations with the method specified in sections 5.2 and 5.3 above. By averaging over sufficiently many stochastic runs of the simulation, we can average the elements of the pure state density matrices for each run and obtain an estimate of the result of the noisy evolution. In this case we are primarily interested in the probability that Shor's algorithm will succeed, and in doing so produce an output which is useful for factorising a number N. The problem of which coefficients are relevant to this measurement of success is addressed in [129], which summarises the procedure of converting a measured value into a value of r and concludes that the useful measurement values are $j_{c_1} = \lfloor c2^{2L}/r \rfloor$ and $j_{c_2} = \lceil c2^{2L}/r \rceil$ for each integer $c \leq r$. As such, after performing simulations we only need to calculate the elements of the density matrix $\rho_{j_{c_1}j_{c_1}}$ and $\rho_{j_{c_2}j_{c_2}}$ for each c. The sum of these elements tells us to the probability of obtaining a useful output to the algorithm.

We performed simulations of Shor's algorithm with Z errors with a variety of N values between 4 and 60 and every possible r value for each N value. To obtain each probability we averaged the probability of success over the results of 750 independent stochastic simulations. This number of stochastic simulations seemed to give appropriate results for the range of (N, a) values tested. We also performed larger simulations of up to $N = 183$ (24 qubits) in which each probability was the average of results obtained from 100 stochastic simulations. Erroneous results were obtained with this number of runs in which an exponential decline of success probability would be interspersed with probabilities of 0 (up to the limits of computational accuracy). As this occurred for (N, a) values which displayed no such artifacts when averaged over 750 simulations, we conclude that 100 is an insufficient number of stochastic simulations to average over to obtain a reasonable result in this range of (N, a) values.

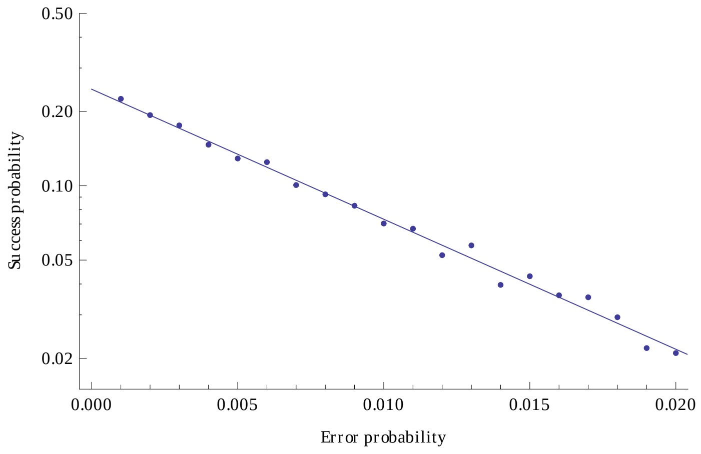


Figure 5.3: The probability of success of Shor's algorithm for $N = 45$ and $a = 17$, which gives $r = 4$, for various error probabilities per time step. Also shown is the line of best fit to this data.

We display our results of the probability of success of Shor's algorithm for $N = 45$ and $a = 17$ (giving $r = 4$) at various probabilities of error per time-step p between 0.001 and 0.02 in figure 5.3. It is clear that the probability of success of the algorithm decreases exponentially with increasing error probability. In figure 5.3 we also show the line of best fit the data. Writing $p_s = ae^{-cp_e}$ where p_s is the success probability and p_e is the error probability per time-step, this gives us a decay of the probability of success $c = 121 \pm 3$.

We obtain similar decay curves to figure 5.3 for each tested value of (N, a). The results of performing a linear fit to the exponential decay of each curve are shown in figure 5.4 where we plot the decay rate c against the value being factorised N. Simulations were performed for each possible a value for each N value, resulting in decay rates for multiple r values for each N. Note that we fit to all of the data for each (N, r) pair which in general may include more than one a value. To a good approximation the operation of Shor's algorithm depends only upon the value of N and r and so neglecting the differences between evolutions with different a values but the same r values simplifies the analysis of our results. It is apparent from

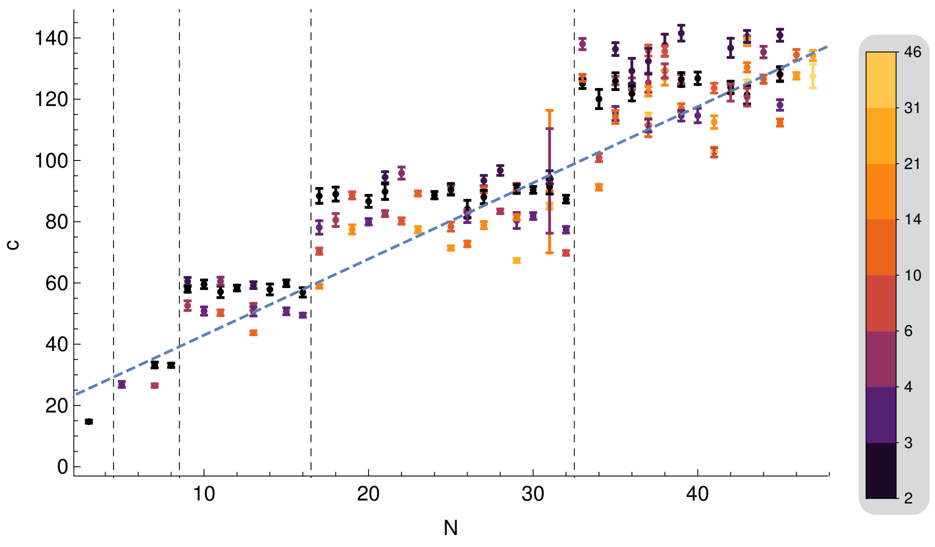


Figure 5.4: The decay rates (c) of the probability of success of Shor's algorithm with increasing probabilities of Z errors on each qubit at each time step. Dotted lines divide results with different numbers of qubits. The colour of each data point encodes the value of r in the corresponding simulation. The line of best fit is shown as a blue dotted line.

figure 5.4 that the decay rate is principally determined by how many qubits $n = \lceil \log_2(N) \rceil$ are used in the simulation. The number of qubits directly determines the number of gates and the number of errors and so we would expect this to be the case. The decay rate appears to be dependent to a lesser degree on the values of N and r; however we can detect no trend in the size of c as N is varied (with n the same).

We also show the line of best fit to the data in figure 5.4, which specifies a relationship between N and c of $c = (18 \pm 2 + (2.49 \pm 0.09)N) f(a, r)$ where $f(a, r)$ is an unspecified function. This line of best fit does not seem to adequately capture the behaviour of the data in figure 5.4, although the clustering of c values for each n makes a more concrete statement difficult to make. As the value of c is principally determined by n rather than N, we might expect c to depend upon $\log(N) \sim n$ rather than N. Such an analysis is suggestive of a logarithmic plot, and so in figure 5.5 we show c plotted against N on a log-log scale. The linear relationship here indicates that $\log(N) = \alpha + \beta \log(c)$ or equivalently $c = AN^b$. The gradient of this line of best fit gives $b = 0.69 \pm 0.03$. This linear fit appears visually to better reflect the data. However, as noted above it is difficult to extend this into

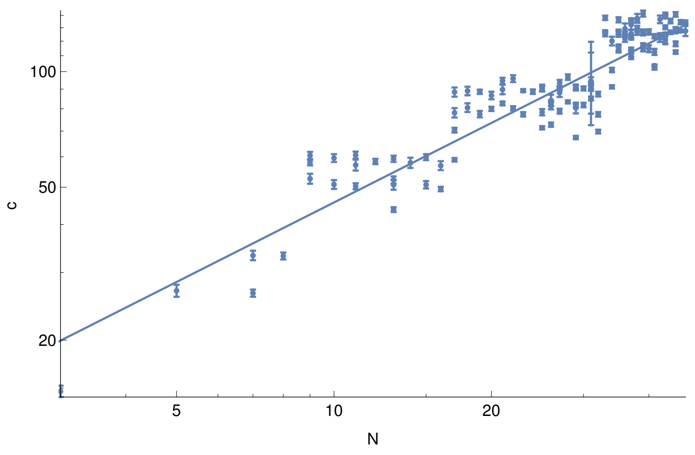


Figure 5.5: The decay rate of the probability of success of Shor's algorithm with increasing probabilities of Z errors on each qubit at each time step plotted on a log-log scale, as well as the line of best fit.

a more rigorous statement. To do so, we would need to determine the c value for larger N values which correspond to larger numbers of qubits in the simulation.

It appears broadly from figure 5.4 that the value of c decreases as r increases, but there are numerous cases in which this is not the case and few N values have enough corresponding values of r to draw concrete conclusions. To elucidate the relationship between N and r we plot the values of c against r in figure 5.6 for all cases in which $n = 5$ and $n = 6$ and for which r is even (figure 5.4a) and odd (figure 5.4b) We note that c is the lowest (and the decay slowest) when r is a power of two, and c decreases with increasing powers of 2. Additionally, c is higher for odd values of r than for even values. Otherwise, there does not seem to be any trend between increasing values of r and the value of c, although more data would be needed to confirm this statement. To explain the behaviour of c as r is varied we need to know where the entanglement between different qubits is concentrated in Shor's algorithm, which we will explain in the next section. As such, we will return to the explanation of this behaviour in section 5.5.1.

5.5 Entanglement in Shor's algorithm

We will develop the notion of how much entanglement is present in Shor's algorithm by considering the different kinds of values which r can take. We

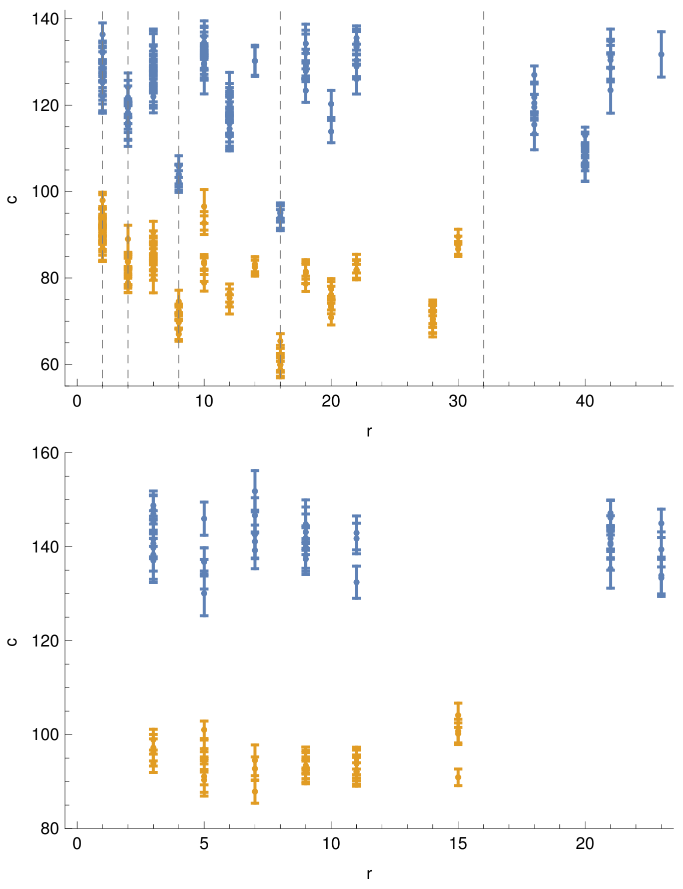


Figure 5.6: All decay rates c of the probability of success of Shor's algorithm with increasing probabilities of error per qubit per time step for all N values with $n = 5$ (yellow curves) and $n = 6$ (blue curves) with (a) even values of r and (b) odd values of r. Values of r which are a power of two are shown with dotted lines in (a).

first consider values of r which are powers of two. In this case Kendon and Munro [142] (henceforth KM) found that only the first qubit in the upper register is entangled with the lower register when $r = 2$ and the first two qubits in the upper register with the lower register when $r = 4$. They also found that there is no pairwise entanglement within either register (quantified by the concurrence) when $r = 2$.

We write the co-prime of N as y and so the modular exponentiation gate is $U^j|x\rangle = \big|(xy^j) \mod N\big\rangle$. In this case we write the period of the modular exponentiation procedure as $r = 2^a$ and so $U^{2^a} = I$. The modular exponentiation part of Shor's algorithm applies gates in increasing powers of 2 and so the jth controlled gate is $U^{2^{j-1}}$. If we set $j > a$ then $U^{2^{j-1}} = \left(U^{2^a}\right)^{2^{j-1-a}} = I$. All modular exponentiation gates after the ath gate are thus the identity. All gates before this will be a permutation of the two permissible values in the lower register. As such, the first a gates will entangle their control qubits with the lower register but the remaining gates will not. As we swap the first control qubit to the top position in the upper register and so on, this will result in the top a qubits in the upper register being entangled with the bottom register, but none of the other qubits in the upper register displaying any entanglement with each other or the bottom register. This explains and generalises the first of KM's observations.

We now consider the pairwise entanglement between individual qubits in the same register when r is any power of two. In this case there are r permissible values in the lower register which are entangled with the r states in the first a qubits of the upper register but not with any others. It is clear that the $2l-a$ bottom qubits in the upper register are separable and so have no pairwise entanglement between themselves or with other qubits in the top register. The remaining a qubits encode r different computational basis states which are entangled with r orthogonal states in the lower register. Tracing out the lower register and all except two of the first a qubits in the upper register will then produce the mixed state $\rho = 1/4(|00\rangle\langle 00| + |01\rangle\langle 01| + |10\rangle\langle 10| + |11\rangle\langle 11|)$. The concurrence of this fully mixed state is zero. Similarly, to consider the reduced density matrix of any two qubits in the lower register we note that each of the basis states in the upper register is orthogonal and so the reduced density matrix will take the form $\rho_r = a/4|00\rangle\langle 00| + b/4|01\rangle\langle 01| + c/4|10\rangle\langle 10| + d/4|11\rangle\langle 11|$. This mixed state has zero concurrence. As such, there is no pairwise entanglement between any two states in the bottom register or in the top register when r is a power of two, proving and generalising KM's second observation. We note that our argument about the reduced state of the lower register can be straightforwardly generalised to any period r, showing that during Shor's algorithm there will never by any pairwise entanglement in the lower register.

We now consider r values which are odd. In this case, it is clear that the upper register and the lower register are entangled. However, the periodic structure of the entanglement of the upper and lower registers cannot be localised to any individual qubits in the upper register. Rather, the whole upper register is entangled with the permissible states in the lower register.

As such, tracing out any qubit in the upper register will always decrease the entanglement between the upper and lower registers. Tracing out more qubits in the upper register will in turn lead to a mixed state displaying even less entanglement between the upper and lower registers. An implication of this is that in the large N limit with small r, there will be no entanglement between any single qubit from the upper register and the lower register. We illustrate this in figure 5.7 which shows the log negativity between any of the first a qubits in the upper register and the lower register after time step a of the modular exponentiation with $N = 1898, a = 1389, r = 3$. The first modular exponentiation gate fully entangles the first qubit with the lower register, but each subsequent gate decreases the log-negativity of any previously used qubit with the lower register.

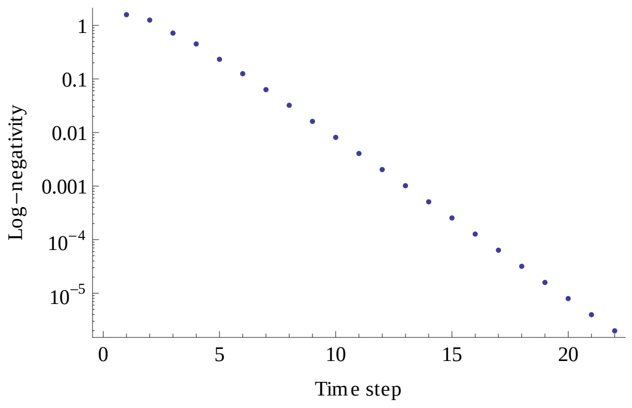


Figure 5.7: The log-negativity between any of the first a qubits in the upper register and the lower register of Shor's algorithm during modular exponentiation for the factorisation of $N = 1898$ and $a = 1389$ giving $r = 3$.

We now consider r values which are even, but which are not powers of two. In this case we can write $r = b \cdot 2^a$ where b is odd. We divide the upper and lower registers into the last a qubits and the other qubits. To simplify our notation, We also consider the permissible states in the bottom register to be $\{|0\rangle, |1\rangle, \ldots, |r\rangle\}$. Up to a normalization constant, taking $2n$ qubits in the upper register, we can write the periodic state formed by the modular exponentiation process as:

$$\begin{aligned}&|0,0\rangle|0,0\rangle + |0,1\rangle|0,1\rangle + \ldots + |0,2^a\rangle|0,2^a\rangle + |1,0\rangle|1,0\rangle + \ldots \\ &\quad + |b,2^a\rangle|b,2^a\rangle + |b+1,0\rangle|0,0\rangle + |b+1,2^a\rangle|0,2^a\rangle + \ldots \end{aligned} \tag{5.5}$$

We can see from (5.5) that the last a qubits of the upper register and the last a qubits of the lower register are maximally entangled. The remaining qubits in the upper and lower register are entangled, but to a lesser degree than the last a qubits in each register. This remaining entanglement will be multipartite across all remaining qubits in the upper and lower registers, as we discussed for the odd r case. There will be no entanglement between any of the first $2n - a$ qubits in the upper register and the lower register in the large N limit. Our way of writing the lower register here is not fully general. However, we can perform a unitary permutation to the whole lower register of (5.5) and recover a state with a general lower register. As a unitary operation on the lower register cannot change the entanglement between the upper and lower registers, our conclusion that the last a qubits of the upper register are highly entangled with the lower register is fully general. Similarly, the first $2n - a$ qubits in the upper register are always entangled with the lower register, but to a lesser extent than the last a qubits and with the entanglement spread across all of these upper register qubits. Note that the qubit ordering we have considered here in the upper register is different to the ordering we usually consider. That is, we usually consider (as in our discussion of entanglement with r a power of 2) the upper register to be reversed.

5.5.1 The decay of success probability with different r values

We are now in a position to explain figure 5.6, which shows how c varies with r. To explain this behaviour, we assume that decoherence which is more destructive to the entanglement of the state will be more destructive to the usefulness of the algorithm. We note that while this assumption produces the correct implications for the variation of c with r, it is still an assumption.

The value of c, and so the rate of decay of the probability of success of Shor's algorithm, is lowest for r values which are a power of 2. If $r = 2^a$, then only the first a qubits are entangled with the lower register. As such, errors occurring on the remaining qubits do not impact on the entanglement of the state. We would expect that this would decrease the rate of decay of probability of success, as we have observed.

From figure 5.6 we can conclude that c is lower for even r values than for odd r values. With reference to the entanglement in the algorithm, we note that for even r, most of the entanglement is concentrated in a subset of the qubits in the upper register. Errors uniformly distributed throughout the algorithm will thus have be less destructive to the entanglement of the state in this case than if the entanglement was completely multipartite between all upper and lower register qubits. As such, we would expect that the algorithm would be more robust with even values of r than odd values of r, as we observe.

Finally we note that when r is odd, the value of c appears to remain roughly constant independent of the value of r. Following on from our analysis of the entanglement present in Shor's algorithm, for odd values of r the entanglement between the upper and lower registers is distributed across the entire upper register regardless of the value of r. The upper register is twice as large as the lower register and so the majority of errors will affect qubits in the upper register. Additionally, the lower register is sparsely populated with permissible states. As such, we would expect that the decline in entanglement between the registers would be relatively constant with different odd values of r. We observe such an effect in figure 5.6 for the value of c as r is varied.

5.6 Discussion

In this chapter we have extended the MPS simulation technique of WHH to include Z errors occurring anywhere in the simulation. Our techniques mostly concern the isolation of parts of the computation that are repeated in the same way every time the computation is performed. In this way, we minimise the time required to perform multiple stochastic simulations of the algorithm with different errors occurring at different times and in different locations. In this context, all Z errors can be moved out of the QFT and the operation of the QFT is the same every time it is performed. We thus recover a motivation for our work on characterising a tensor network representing the QFT in chapter 4.

While it is possible to perform simulations of large instances of Shor's algorithm with Z errors using our techniques, the same methods cannot be applied to other error models. Most notably, X errors occurring during the modular exponentiation step of the algorithm quickly cause a highly entangled state which is not easy to represent with a MPS. It is not clear how a MPS simulation of Shor's algorithm could be performed with such noise occurring. For general noise models, a state vector or density matrix simulation, such as those that are usually developed to simulate Shor's algorithm, may have to be employed.

Different error models are used in each past noisy simulation of Shor's algorithm. Because of this, as well as the different approaches taken to performing the simulation and interpreting the results, it is difficult to directly compare the results obtained. Broadly (although with exceptions noted in section 5.1.3), past work has found that with increasing error size, the success of Shor's algorithm decreases exponentially. We obtain such a result, with the added issue that the decay rate of the algorithm appears to increase linearly with the number being factorised, and so exponentially with the number of qubits. In light of this result, an increase in the size of the algorithm would cause a sizable drop in the effectiveness of the algorithm with even a small probability of errors occurring. Such a scaling of the decay rate bodes very poorly for the scalability of Shor's algorithm without a fully fault-tolerant quantum computer.

While in this chapter we have described techniques by which very large simulations of Shor's algorithm may be performed, the simulations we have performed have been relatively modest in size. Our simulations were performed on a single core of a laptop computer and serve mainly as a proof of principle of our techniques rather than a full exploration of their use. Our simulations were exhaustive (we simulated all possible co-primes for every value of N) and it is probable that further time savings could be found in future simulations. In particular, our simulations used the full MPO of the QFT, while if only a small number of output coefficients were considered the QFT can be converted into a series of bond size one projectors (this point is noted in section 5.2). It would also be worthwhile to consider simulating Shor's algorithm with a single control qubit, as has enabled the large simulations of [151], or with classical modular exponentiation, as in [139]. We are confident that with these improvements, and access to more computational power, our techniques may be employed for very large simulations.

Chapter 6

Simulation of Grover's Algorithm

In this chapter, we will use MPSs and MPOs to simulate Grover's algorithm. Grover's algorithm has been simulated in the past using a variety of methods. Most simulations take place either in a specialised reduced subspace (for example that in which Grover's algorithm was described in section 2.2) or with a straightforward state vector simulation. Large state vector or parallel simulations of Grover's algorithm include [135, 137, 153, 154]. A notable exception is the work of Viamontes et al [155, 156, 157], who develop a simulation technique using a data structure called a Quantum Information Decision Tree, which is an extension on data structures used to perform simulations of classical logic circuits. This structure is used to perform classical gate simulations of Grover's algorithm with with one solution and 40 qubits in under 23 hours. There has also been prior work on using tensor networks to simulate quantum search. Kawaguchi et al [158] perform a tensor network based DMRG simulation of Grover's algorithm for up to 35 qubits In [159] a MPS is used to simulate a Bruschweiler search, which is a quantum search taking place in a bulk ensemble quantum computer. Finally, an example program showing the simulation of Grover's algorithm with a MPS is given in [160].

Many papers have been published which simulate Grover's algorithm in the presence of various kinds of noise and calculate the affect on the success of the algorithm. In general these papers perform simulations of limited sizes explicity storing the state vector or density matrix. Some exceptions include

[161] which takes advantage of symmetries in the density matrix to perform large simulations and [162, 163] which uses perturbation theory to calculate the probability of success of Grover's algorithm with dephasing noise of a small size. Many different error models are considered including imperfect gates [164], unitary imperfections in the Hadamard operators [165, 166], phase noise [167, 168, 162, 163], noise in the oracle [169, 168], dissipative noise [170, 89], depolarising noise [171], static inter-qubit interactions [172] and physical noise models based on realistic architectures or external fields [173, 174, 175]. In general past works have found that there is an exponential decrease in the probability of success with increasing noise levels, and in the work of Zhirov [170] and Salas [171], on the number of qubits. Several papers find a scaling for the allowable noise if Grover's algorithm is still to outperform classical search of $\epsilon = N^{-a}$, where ϵ is the size of the noise and the value of a depends upon the study and the noise model used [167, 166, 169, 171].

In section 6.1 we will summarise the circuit decomposition of Grover's algorithm and in section 6.2 describe how such a decomposition can be used to simulation Grover's algorithm with MPOs. We detail a technique to determine the solution to a search problem given the MPO of its oracle in section 6.3 and discuss the implications of this technique for the computational capability of the algorithm. We perform simulations of Grover's algorithm in section 6.4 and determine how the success of the algorithm scales with increasing levels of noise and numbers of qubits.

6.1 Quantum circuit for Grover's algorithm

In order to represent Grover's algorithm an a MPO, we need to write the algorithm in a circuit format. As detailed in section 2.2, Grover's algorithm can be performed in the following stages:

1. Prepare a state $|\psi_0\rangle = |0\dots0\rangle$ and perform a Hadamard gate on each qubit.

2. Complete the Grover iteration r times. Each iteration consists of:

   (a) Apply the oracle $\mathbf{O}$, $|i\rangle \rightarrow (-1)^{f(i)}|i\rangle$.

   (b) Apply the inversion about the mean operator $2|\psi\rangle\langle\psi| - \mathcal{I}$.

3. Measure each of the qubits. The result is likely to be a solution to the search problem.

To convert this prescription into a quantum circuit, we consider the inversion about the mean. As we know that $|\psi\rangle = H^{\otimes n}|0\rangle$, and $H^{\otimes n} I H^{\otimes n} = I$, so we can rewrite the inversion about the mean:

$$2|\psi\rangle\langle\psi| - I = H^{\otimes n}\left(2|0\rangle\langle 0| - I\right) H^{\otimes n} \tag{6.1}$$

The term $(2|0\rangle\langle 0| - I)$ is equivalent up to a global phase to an operator which applies a -1 phase to $|0\rangle$, but not to any other computational basis state. The operator $2|0\rangle\langle 0| - I$ can be further decomposed into a universal set of gates such as one including H, X and CNOT. We show a decomposition involving H, X and a Toffoli gate in 6.1. Note that a n qubit Toffoli gate can be easily decomposed into a series of CNOT gates.

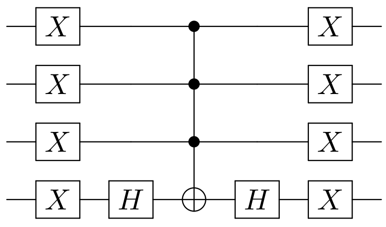


Figure 6.1: A quantum circuit for the operator $-2|0\rangle\langle 0| + I$ for four qubits, which is equivalent up to a global phase to $2|0\rangle\langle 0| - I$.

Combining this gate decomposition with the general prescription of Grover's algorithm outlined above allows us to write Grover's algorithm as a quantum circuit. A quantum circuit for four qubits is shown in figure.

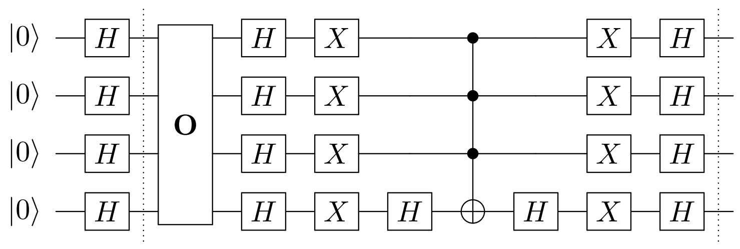


Figure 6.2: A quantum circuit to perform Grover's algorithm with 4 qubits. The dotted lines enclose the Grover iteration, which must be repeated multiple times to obtain the solution with a high probability.

6.2 Simulation with a MPO

It is possible to take the circuit described above, select a specific oracle and decompose both this oracle and the n qubit Toffoli gate into individual CNOT gates and then to perform a MPO circuit simulation as described in section 3.4. However, it is easier to directly construct MPOs which encode the inversion about the mean operator and the oracle, and then to use these MPOs together with the other circuit elements to simulate Grover's algorithm. We will detail our full simulation technique and its time scaling in section 6.2.5, after first describing the construction of the relevant MPOs for parts of the quantum circuit in sections 6.2.1 and 6.2.2 and the complexity of the oracle MPO in section 6.2.3.

6.2.1 A MPO encoding inversion about the mean

It has previously been shown [176] that there is an equivalence between MPSs and MPOs and finite state weighted automata. As such, if an automaton can be written that produces an operator, it can be converted into a tensor network describing it. We use this technique to obtain an operator for inversion about the mean. It is straightforward to apply one qubit gates to a MPO and so from (6.1), we note that we only need to produce a MPO encoding the operator $2|0\rangle\langle 0| - I$. To do this, we will write a simple algorithm which produces the coefficient $|i\rangle\langle j| = |i_1, \ldots, i_n\rangle\langle j_1, \ldots, j_n|$ in this operator:

```
a ← 0
**for** each qubit $k$ **do**
    **if** $i_k \neq j_k$ **then return** 0
    **end if**
    **if** $a = 0$ and $i_k = 1$ **then** $a \leftarrow 1$
    **end if**
**end for**
**if** $a = 0$ **then return** 1
**elsereturn** $-1$
**end if**
```

We can draw connections between this algorithm and a finite state automata by noting that the variable a encodes the state of the system and can take two values: 0 and 1. We additionally have the variable k which encodes which qubit is being addressed. We could thus write this algorithm as a finite state automata where a is the state of the system and there are k individual decision points, each corresponding to one qubit. Such a finite state automata can be translated into a MPO by using the state of the system (a) as the bond index and encoding the decision structure of the automata into the tensors. Doing so provides us with a set of tensors:

$$\begin{aligned} \Gamma^{[0]0}_{ij} &= \delta_{ij}\delta_{i0}, & \Gamma^{[0]1}_{ij} &= \delta_{ij}\delta_{i1}, \\ \Gamma^{[k]00}_{ij} &= \delta_{ij}\delta_{i0}, & \Gamma^{[k]11}_{ij} &= \delta_{ij}\delta_{i1}, \quad k \neq n, \\ \Gamma^{[n]0}_{ij} &= -\delta_{ij}\delta_{i0}, & \Gamma^{[n]1}_{ij} &= \delta_{ij}\delta_{ij}, \end{aligned} \tag{6.2}$$

where all other elements are zero. We set all bonds to be evenly weighted across all indices: $\lambda^{[k]} = (1, 1)$. This MPO is not in canonical form, but we can easily transform it to canonical form by doing a series of SVDs.

6.2.2 MPOs encoding arbitrary oracles

To elucidate the construction of a MPO of any oracle we first consider consider the construction of a MPO encoding an oracle with a single solution. We define a set of functions $\{f_k(i)\}$ which return 0 if i is not the kth digit of the solution and 1 if it is. In this case the algorithm to determine the $|i\rangle\langle j| = |i_1, \ldots, i_n\rangle\langle j_1, \ldots, j_n|$ coefficient will be exactly the same as the one described in section 6.2.1, but instead of testing if $i_k = 1$, we test if $f(i_k) = 0$, and we swap the return values. We can then follow the same procedure to obtain a series of tensors encoding the oracle:

$$\begin{aligned} \Gamma^{[0]} &= \left(\begin{bmatrix} f_0(0) & 0 \\ 0 & f_0(1) \end{bmatrix} \begin{bmatrix} 1 - f_0(0) & 0 \\ 0 & 1 - f_0(1) \end{bmatrix} \right), \\ \Gamma^{[k]} &= \begin{pmatrix} \begin{bmatrix} f_k(0) & 0 \\ 0 & f_k(1) \end{bmatrix} & \begin{bmatrix} 1 - f_k(0) & 0 \\ 0 & 1 - f_k(1) \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} & \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \end{pmatrix}, \\ \Gamma^{[n]} &= \begin{pmatrix} \begin{bmatrix} f_n(0) & 0 \\ 0 & f_n(1) \end{bmatrix} \\ \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \end{pmatrix}. \end{aligned} \tag{6.3}$$

We use $\lambda^{[k]} = (1, 1)$ for each of the vectors.

To compute a MPO for an oracle of arbitrarily many solutions, we compute an oracle using (6.3) for the first solution, and then a MPO for each additional solution which encodes an operator of the form $-2|s\rangle\langle s|$, where s is the solution. This MPO has only a single non-zero coefficient, so it is easy to write a series of tensors which generate it:

$$\Gamma^{[k]}_{i_k j_k} = \delta_{i_k j_k}\delta_{f(i_k)\,1}, \quad \lambda^{[k]} = (1) \quad \text{for all } k, \tag{6.4}$$

where we have implicitly set all bonds to size one, and so have not notated them. By adding the first solution's MPO with each of the remaining MPOs and performing a series of SVDs each time, we will obtain a canonical MPO representation of a oracle with an arbitrary number of solutions.

6.2.3 The complexity of the oracle MPO

To simulate Grover's algorithm, it is necessary to select a specific subspace of computational basis states to be the solution to the search problem, and then to construct an operator to perform the required oracle operation in the algorithm. In this context, it is worthwhile to ask how complex the MPO representation of the oracle is for different solution subspaces. We additionally ask what implications this has for the complexity of Grover's algorithm and when it would provide a speedup over doing a classical computation.

If we have n qubits, the Hilbert space in which the search takes place is $\mathcal{H}_2^{\otimes n}$ while if there are m solutions, the solutions will be located in a m dimensional subspace of the larger Hilbert space. The oracle in this case is an operator which applies a phase of -1 to the coefficient of each element in the solution subspace but does not affect the other coefficients. We will consider a function $f(x)$, which when given the index of a computational basis state in the overall Hilbert space, returns 1 if the state is a solution to the search problem and 0 otherwise. In this case, we may write the oracle as:

$$\hat{O} = \sum_{i=1}^{2^n} (-1)^{f(i)} |i\rangle\langle i| \tag{6.5}$$

We can then consider a bipartition of this operator into two equally sized halves $l|r$. The Hilbert space of each of these partitions will then have dimension $d = 2^{n/2}$ and so we may write

$$\hat{O} = \sum_{l,r=1}^{d} (-1)^{f(ld+r)} |l\rangle\langle l| \otimes |r\rangle\langle r| \tag{6.6}$$

We can write this operator using tensor notation across this bipartition:

$$\hat{O} = \sum_{i_l,i_r,j_l,j_r=1}^{d} O^{i_r j_r}_{i_l j_l} |j_l\rangle\langle i_l| \otimes |j_r\rangle\langle j_l|, \quad O^{i_r j_r}_{i_l j_l} = \delta_{i_l j_l}\delta_{i_r j_r}(-1)^{f(i_l d+i_r)} \tag{6.7}$$

We can write this tensor in a matrix, where the rows encode the operator in the left partition whose coefficient is being described, and the columns the operator in the right partition:

$$O = \begin{bmatrix} (-1)^{f(1)} & 0 & \cdots & 0 & (-1)^{f(2)} & 0 & \cdots & \cdots & 0 & (-1)^{f(d)} \\ 0 & & & & 0 & & & & & 0 \\ \vdots & & & & \vdots & & & & & \vdots \\ 0 & & & & 0 & & & & & 0 \\ (-1)^{f(d+1)} & 0 & \cdots & 0 & (-1)^{f(d+2)} & 0 & \cdots & \cdots & 0 & (-1)^{f(2d)} \\ 0 & & & & 0 & & & & & 0 \\ \vdots & & & & \vdots & & & & & \vdots \\ \vdots & & & & \vdots & & & & & \vdots \\ 0 & & & & 0 & & & & & 0 \\ (-1)^{f((d-1)d+1)} & 0 & \cdots & 0 & (-1)^{f((d-1)d+2)} & 0 & \cdots & \cdots & 0 & (-1)^{f(d^2)} \end{bmatrix} \tag{6.8}$$

This matrix is clearly similar to another matrix in which we eliminate all rows and column which have no non-zero elements.

$$O \sim \begin{bmatrix} (-1)^{f(1)} & (-1)^{f(2)} & \cdots & (-1)^{f(d)} \\ (-1)^{f(d+1)} & (-1)^{f(d+2)} & \cdots & (-1)^{f(2d)} \\ \vdots & \vdots & \ddots & \vdots \\ (-1)^{f((d-1)d+1)} & (-1)^{f((d-1)d+2)} & \cdots & (-1)^{f(d^2)} \end{bmatrix} \tag{6.9}$$

The new matrix (6.9) has d rows and d columns as we have eliminated all elements which encode coefficients for operators $|i\rangle\langle j|, i \neq j$ in each partition. The rank of a matrix is the number of linearly independent columns or rows that it contains, and so the rank of the matrix (6.9) will be one greater than the minimum of the number of rows and the number of columns that contain at least one -1. As we have written the matrix (6.9) by differentiating between the left and right partitions of the Hilbert space, the rank of the matrix is the Schmidt number of the operator O. For example, if there is one solution to the search problem, the Schmidt number of oracle will be 2. If there are two solutions to the search problem, the Schmidt number of the oracle can be either 2 or 3. As (6.9) has $d = 2^{n/2}$ rows and columns, the maximum Schmidt number of the oracle will be $2^{n/2} + 1$. By comparison, the maximum Schmidt number of an operator on n qubits is $N = 2^n$, and so the oracle has a Schmidt number of approximately $\sqrt{N}$.

6.2.4 Using two MPOs for simulation

When discussing the computational complexity of Grover's algorithm and its entangling properties, it is useful to construct an operator using the MPOs for inversion about the mean and the oracle described above, as well as the appropriate one qubit gates. However, it is difficult to construct an operator in this way to simulate the algorithm on many qubits. To see why this is the case, we consider the action of a single Grover iteration on a computational basis state $|\phi\rangle$. A randomly selected state is very unlikely to be a solution of the search problem, so the oracle will most likely act as the identity on $|\phi\rangle$. The inversion about the mean will act on $|\psi\rangle$ in the following manner:

$$\left(2|\psi\rangle\langle\psi| - \hat{I}\right)|\phi\rangle = 2\langle\psi|\phi\rangle|\psi\rangle - |\phi\rangle. \tag{6.10}$$

As $|\psi\rangle$ is an equal superposition of all basis states, $\langle\psi|\phi\rangle$ will be small and so the overall action of a Grover iteration on $|\phi\rangle$ is to a good approximation that of the identity operator. A similar conclusion can be drawn for an operator composed of many Grover iterations. This presents a complication for MPO based simulation as the operator of a series of Grover iterations will only slightly different from the MPO of the identity operator. This will lead to singular value vectors in the bonds of the MPO which contain one very large number (almost $2^{n/2}$ where n is the number of qubits) and a series of much smaller values. As the number of qubits is increased, the larger value will dominate the smaller values and so the condition number of the SVDs used in simulation will increase. We would thus need to use high precision numbers to perform effective simulations with many qubits.

By constructing MPOs of the full Grover operator for several sizes and numbers of Grover iterations, we find that the largest singular value is approximately $2^{n/2}$, as expected, while the smaller singular values remain a roughly constant size as the number of qubits is increased. The condition number of SVD problems will thus increase exponentially with the number of qubits. Allowing for the number of operations performed to form a MPO and for the sizes of the smaller singular values to vary by a few orders of magnitude, this will quickly cause the computation of the MPO to become difficult to perform.

This issue can be resolved by noting that Grover operators differ only slightly from identity operators, and so there is only one large singular value in MPOs encoding Grover operators. As such, we can reduce the MPOs we are forming into a sum of a separable operator and a much smaller deviation operator:

$$\hat{U} = \hat{U}_s + \hat{U}_d \tag{6.11}$$

We now outline how to perform operations on these separable deviation operators (henceforth SD-MPOs). In Grover's algorithm we are required to perform one qubit unitaries, find the products of SD-MPOs and finally apply SD-MPOs to MPSs. By explicitly constructing MPOs representing the oracle and inversion about the mean operators, we do not need to perform any two-qubit unitary gates.

To apply a one qubit gate $\hat{V}$ to a SD-MPO, we just apply it to both constituent operators

$$\hat{V}\hat{U} = \hat{V}\hat{U}_s + \hat{V}\hat{U}_d. \tag{6.12}$$

To apply one SD-MPO to another, we perform several operator-operator products:

$$\left(\hat{V}_s + \hat{V}_d\right)\left(\hat{U}_s + \hat{U}_d\right) = \hat{V}_s\hat{U}_s + \left(\hat{V}_s\hat{U}_d + \hat{V}_d\hat{U}_s + \hat{V}_d\hat{U}_d\right). \tag{6.13}$$

As both V_s and U_s are separable, $\hat{V}_s\hat{U}_s$ is also separable. Additionally, as separable operators, V_s and U_s only codify a preferred basis. We would not expect one qubit unitaries to change the size of a small operator, and so

$\hat{V}_s\hat{U}_d$ and $\hat{V}_d\hat{U}_s$ are both small deviations from the separable operator. As the product of two small deviation operators, $\hat{V}_d\hat{U}_d$ is also a small operator. As such, (6.13) is in the form of (6.11). To apply a SD-MPO to a MPS, we first apply the separable operator, and then the deviation operator, and then add the resulting MPSs. We may also contract both of the resulting MPSs when a coefficient is required and then add the resulting coefficients.

In order to construct a SD-MPO of Grover's algorithm, we consider the oracle as the sum of the of the identity operator and a deviation for each search solution $a \in \{s\}$:

$$\hat{O} = \hat{I} - 2 \sum_{a\in\{s\}} |a\rangle\langle a|. \tag{6.14}$$

We can form this SD-MPO for one solution as outlined in section 6.2.4. It is straightforward to generalise to oracles with larger number of solutions.

We consider the inversion about the mean to be formed by the quantum circuit shown in figure 6.1, and so to create a SD-MPO for this process we need to form a SD-MPO representing the Toffoli gate, and then apply local one qubit gates to it. The separable part of the Toffoli gate is the identity and the deviation is the operator $|1\dots1\rangle\langle1\dots1| \otimes \left(\hat{X} - \hat{I}\right)$. We can create this deviation MPO with the tensors:

$$\Gamma^{[k]}_{i_kj_k} = \delta_{i_k1}\delta_{j_k1},\ k \neq n, \qquad \Gamma^{[n]}_{i_nk_n} = (X - I)_{i_nj_n} \tag{6.15}$$

6.2.5 Simulation technique

We simulate a single run of Grover's algorithm with n qubits, a given set of solutions $\{s\}$ and m Grover iterations by constructing a SD-MPO of the overall operator of the algorithm. We construct this SD-MPO in several steps.

1. Construct a SD-MPO for a single Grover iteration. To do this we break the Grover iteration into an oracle call and inversion about the mean. We have detailed in sections 6.2.1, 6.2.2 and 6.2.4 above how to construct these SD-MPOs. The oracle SD-MPO can then be applied to the inversion about the mean SD-MPO, interspersed with the appropriate one qubit gates.

2. We construct a series of SD-MPOs encoding the Grover iteration op-

erator raised to increasing power of two, up to the power of two whose value is directly below m:

$$\hat{G}^{2^0}, \hat{G}^{2^1}, \hat{G}^{2^2}, \ldots, \hat{G}^{2^{\lfloor \log_2 m \rfloor}}. \tag{6.16}$$

In this series, each operator is the square of the previous operator and so each SD-MPO can be formed by applying the previous SD-MPO to itself.

3. We construct a SD-MPO encoding the correct number of Grover iterations. To do this, we start with a separable MPO of the n qubit identity and a deviation MPO of 0. For each 1 in a position k of the binary representation of m, we apply the kth operator in the series (6.16). Doing so leads to a MPO representation of m Grover iterations. The preliminary Hadamard gates can now be applied to the start of the operator to complete the SD-MPO.

The SD-MPO resulting from these steps can be applied to the MPS state $|0\ldots 0\rangle$ and a contraction performed to calculate the probability of an outcome from a single run of Grover's algorithm. By constructing a MPO representing Grover's algorithm with a variety of numbers of qubits, numbers of solutions, solutions and numbers of Grover iterations, we found that the maximum bond dimension of the SD-MPO is $(s+1)^2$, where s is the number of solutions.

We now consider the time required to perform this simulation. Constructing a MPO for the first Grover iteration requires us to apply a Toffoli SD-MPO, which will have bond size 2, to an oracle SD-MPO, which will have bond size s if s is small (this bond size is obvious from (6.14)). This operation will take time $O(s^2 n)$, where n is the number of qubits. We must then produce the list of $\lceil \log_2(m) \rceil$ SD-MPOs by repeated squarings. Each of these SD-MPOs will have bond size of $O(s^2)$ (this is the observed maximum bond size of the SD-MPO with s solutions), so each squaring will take time $O(ns^4)$ for the tensor multiplications, producing tensors with bond size $O(s^4 + 2s^2) \sim O(s^4)$. The series of SVDs immediately after the multiplications will then take time $O(ns^{12})$, as a single SVD will take time $O((s^4)^2 \cdot s^4)$. As we must perform $O(\log(m))$ of these squarings, producing the list of SD-MPOs will take a total time of $O(ns^{12} \log(m))$. The final operator will be formed by starting with the identity and multiplying at most every one of this list of operators by the identity. The SD-MPO formed after each multiplication will have a bond size less than or equal to the maximum observed and this operation will take an amount of time in the worst case of the same order as the formation of the list of SD-MPOs. The amount of time required for the formation of a SD-MPO for the entire algorithm is then $O(ns^{12}log(m)) = O(s^{12}\log(N)\log(m))$.

6.2.6 Simulation technique with errors

We perform simulations with errors in the algorithm immediately after each Grover iteration. We perform simulations with fixed numbers of X, Y and Z errors. The simulations are performed stochastically with discrete errors, and then the results averaged after a large number of random runs.

To perform a single simulation run with n_e errors and m Grover iterations, we produce a random sample of n_e integers less than m. Each integer corresponds to the time step (number of of Grover iterations elapsed) at which an error occurs. By sorting these time steps and taking the differences between adjacent integers, we can find a series of $n_e - 1$ numbers of iterations which take place between errors. We then construct a series of MPOs for each number of iterations. We apply the iteration operators, interspersed with errors at random qubit locations, to the initial MPS. We can finally determine any required coefficients after the application of all operators. We find the probability of success of the algorithm with a given number of errors by keeping a running average of the probabilities of success as we perform individual pure state noisy simulations. Other interesting quantities can be found analogously.

By constructing SD-MPOs for a range of numbers of qubits, numbers of solutions, solutions, numbers of Grover iterations, numbers of errors and types of errors (X, Y Z and a combination of each of these), we find that the presence of errors does not affect the overall bond size of the SD-MPO of the algorithm. This is a surprising result and is in contrast to the results presented in chapter 4, in which introducing small errors in the algorithm was found to have the potential to dramatically increase the bond size. The result that the bond size in Grover's algorithm does not change with the addition of errors can perhaps be explained in the light of a SD-MPO by considering that an error is a one qubit unitary, and so it only changes the preferred basis of the operator in the separable part of the SD-MPO (as well as in the deviation part). Our result indicates that it does not otherwise affect the operation of the SD-MPO.

In light of this result, we now consider the amount of time required to create a SD-MPO of Grover's algorithm with n_q qubits, s solutions, m Grover iterations and n_e errors. We can consider this SD-MPO to be formed from $m+1$ SD-MPOs each of which contains no errors and is created with the method outlined in section 6.2.5. In the worst case the errors will be unevenly spaced throughout the number of time-steps, such that the size of the list of SD-MPOs must be the same size as in the noiseless case with the same number of Grover iterations. In this case, the cost of creating each of the SD-MPOs will be (on average) $O(n_q \log(m) s^{12})$, and as we are forming n_e of these SD-MPOs, the time required to create them will be $O(n_q n_e \log(m) s^{12})$ (we note that if the number of errors is large this scaling will not be accurate as the cost of forming the largest SD-MPO will be amortized across the time of formation of the other SD-MPOs). We must then perform n_e operator-operator multiplications to form the final SD-MPO. Each multiplication will take time (including the SVDs) $O(n_q s^{12})$, and so these multiplications will take a total time of $O(n_e n_q s^{12})$. As such, the overall time for the simulation will be $O(n_e s^{12} \log(N) \log(m))$.

6.3 Determining search solutions from MPOs

In order to show the implications of the oracle bond rank found in section 6.2.3 for the complexity and usefulness of Grover's algorithm, we will only consider the classically most difficult search problem in which there is one solution. In this case, the Schmidt number of the oracle, and its maximum bond rank in a MPO representation, is 2.

If we had access to the MPO representation of the oracle, we could find the solution to the search problem by finding the expectation value of the oracle over a series of separable states. To illustrate this, we could use the following procedure (this method is a modified form of the one presented in [177].

1. On each tensor $\Gamma^{[k]\alpha_{k-1}\alpha_k}_{i_k j_k}$ in the MPO representation of the oracle $\hat{O}$, take the trace over the two quantum state indices to give a new tensor $\Gamma'^{[k]\alpha_{k-1}\alpha_k} = \sum_{i_k} \Gamma^{[k]\alpha_{k-1}\alpha_k}_{i_k i_k}$. This will give a new tensor network with only ancillary indices whose contraction will give the expectation value

$\langle\psi|\hat{O}|\psi\rangle$, where $|\psi\rangle = 1/2^{n/2}\sum_{i=1}^{2^n}|i\rangle$. As there is one solution to the search problem and a total of 2^n possible bit strings, this expectation value will be $1 - 2^{-(n-1)}$.

2. Starting from the rightmost tensor, contract the ancillary indices to compute a series of vectors $\Gamma'^{[k\ldots n]\alpha_{k-1}}$ which encode the contraction of the tensors $\Gamma'^{[k]}\cdots\Gamma'^{[n]}$.

3. On the final tensor (that of the first qubit) $\Gamma_{i_1j_1}^{[1]\alpha_1}$, set $i_1 = j_1 = 0$, giving a vector $\Gamma_{00}^{[1]\alpha_1}$. Multiplying this vector by $\Gamma'^{[2\ldots n]\alpha_1}$ will give the expectation value $\langle\phi|\hat{O}|\phi\rangle$, where $|\phi\rangle = |0\rangle \otimes 2^{-(n-1)/2}\sum_{i=1}^{2^{n-1}}|i\rangle$. This expectation value will either be 0, or higher than the expectation value in step one of $1 - 2^{-(n-1)}$. If it is higher, the first digit of the solution will be 0. If it is 0, there are no solutions in the computational basis states with non-zero coefficients in $|\phi\rangle$, and so the first digit of the solution is 1. We set the state indices of $\Gamma^{[1]}$ to the first digit in the solution and obtain a vector $\Gamma'^{[1]}$.

4. Iterate step three over each of the tensors in the MPO. We will describe this step abstractly for the kth tensor. Set $i_k = j_k = 0$, giving a vector $\Gamma_{00}^{[k]\alpha_{k-1}\alpha_k}$. Contract this vector with the vector from the left $\Gamma'^{[1\ldots k-1]\alpha_{k-1}}$, for which each of the tensors $\Gamma^{[1]}\ldots\Gamma^{[k_1]}$ has had their state indices set to the first $k-1$ digits in the solution. Contract $\Gamma'^{[1\ldots k-1]\alpha_{k-1}}\Gamma_{00}^{[k]\alpha_{k-1}\alpha_k}$ with $\Gamma'^{[k+1\ldots n]\alpha_k}$, giving the expectation value $\langle\phi|\hat{O}|\phi\rangle$ where $|\phi\rangle = |s_1\ldots s_{k-1}\rangle \otimes |0\rangle \otimes 2^{-(n-k)/2}\sum_{i=1}^{2^{n-k}}|i\rangle$ and $|s_1,\ldots,s_n\rangle$ is the computational basis state encoding the solution to the search problem. If this is 0, the kth digit of the solution is 1, otherwise it is 0. Take $i_k = s_k$ to be the kth digit of the solution and set $\Gamma'^{[1\ldots k]} = \Gamma'^{[1\ldots k-1]\alpha_{k-1}}\Gamma_{i_ki_k}^{[k]\alpha_{k-1}\alpha_k}$

After following this procedure, we will have the n bit binary representation of the solution. The only operations which take place in the above algorithm are vector-vector contractions, the number of which scales as $O(n)$ where n is the number of qubits. As the bond size of the MPO representation of the oracle is 2, this algorithm can be performed in $O(n)$ time. The size of the search space is $N = 2^n$, and so the run-time for finding a solution given a MPO of the oracle by MPO contraction is $O(\log(N))$. We note that this algorithm can be straightforwardly extended to higher numbers of solutions, and the same execution time scaling will apply as long as the number of solutions does not grow as the size of the search space grows.

This algorithm, combined with the low rank of the MPO representation of the oracle, highlights the subtlety of the status of the oracle in Grover's algorithm. If an oracle can be accessed quantum mechanically, Grover's algorithm provides a quadratic speedup over the best possible classical search. The results of a specific oracle must be assumed to simulate Grover's algorithm, for example to determine how it performs under the influence of error. However, if we have knowledge of how the oracle functions in MPO format, we can straightforwardly solve the search problem much more easily than by using Grover's algorithm.

This result also prohibits us from producing an oracle for which Grover's algorithm will be useful using many different kinds of quantum circuits. If a circuit is provided to solve an oracle (this can be a quantum circuit or the quantum mechanical analogue of a classical logic circuit), we may start with a MPO representing the identity matrix and then apply the quantum circuit to our MPO, in the process calculating the MPO of the oracle. If Grover's algorithm is to be useful, this classical calculation must be difficult and so must require resources scaling exponentially with the number of qubits. However, we know that the final Schmidt number of the oracle must be 2. As such, the circuit calculating the oracle must produce very large amounts of correlation, leading to a MPO with a high Schmidt number. Interference must then occur between different parts of the system, leading to the required maximal Schmidt number of 2 across every bond. Grover's algorithm cannot be useful for any search problem whose oracle is produced by a quantum circuit which is strictly local. Similarly, a quantum circuit with a logarithmic number of gates in the total size of the Hilbert space cannot produce a large amount of entanglement across the system, and so cannot produce an oracle for which Grover's algorithm would be useful.

In order to gain some insight into which problems Grover's algorithm would provide no speedup for, we consider how long it would take to calculate the MPO of an oracle produced by a quantum circuit of n_g gates. To make the mathematics easier, we will refer to bond height $h = \log_4 d$, where d is the bond size. In a MPO, each tensor adds an additional 4 degrees of freedom with respect to to the previous tensor (these additional degrees of freedom are the local indices of the operator). As such, because the minimum bond size is rank revealing, the bond on each side of the MPO can be at most four times as large as the bond on the other side. This means that the bond height can increase by a maximum of one from neighbouring bonds. This result is a straightforward generalisation of the proof presented in [177], replacing the number of local indices in a MPS (which is 2), with the number of local indices in a MPO (which is 4).

If the goal of a quantum circuit is to generate a bond of as much height as possible, it is clear that at the end of the circuit, the bond heights must increase at a rate of one per qubit from each end of the line of qubits. They must thus form a triangle with the maximum bond height in the middle. If there are n qubits, this triangle will have a maximum height of $\lfloor n/2 \rfloor$. To produce this triangle of bond heights in the most efficient pattern, we start with the identity, in which all bonds are of height 0, and then apply gates to change every second bond to height 1 (there are $\lfloor n/2 \rfloor$ of these gates). Each bond of height 0 surrounded by height 1 bonds must then have a gate applied raising it to height 2. This pattern of gates continues, with each set of gates operating on positions in the spaces between the previous set. Each set will contain one less gate than the previous set, and so a total of the $\lfloor n/2 \rfloor$th triangular number of gates is required to fully entangle a n qubit operator.

The SVD requires time $O(d^3)$ for a two qubit gate with bonds of either side of the two qubits of size d. It is the most computationally demanding part of performing a two qubit gate. As such, using the pattern of gates above, maximally entangling a n qubit MPO will take time:

$$O\left(\sum_{i=0}^{k-1}(k-i)\left(4^i\right)^3\right) \sim O\left(4^{3k}\right), \tag{6.17}$$

where we denote $k = \lfloor n/2 \rfloor$. The number of gates required to produce this entanglement is $n_g = k(k-1)/2$, and so $k \sim \sqrt{2n_g}$ and the time required to produce the entanglement scales as $O\left(4^{3\sqrt{2n_g}}\right)$. This scaling represents the maximum amount of time that performing n_g gates on an identity MPO can take. Note that this circuit may take place on a qubit array which is larger than the number n which we have used in this calculation. In this case, as long as the circuit is (mostly) localized in a region of n qubits, a similar amount of correlation will result and this scaling is accurate. If the total number of qubits is smaller than n, less correlation can be produced and the time taken will be less than that calculated.

We can relate the time scaling we have calculated to the total number of qubits by setting $n_g = n^a$ for some a, in which case we have a calculation time of $O(2^{6\sqrt{2}n^{a/2}})$. If we set a higher than two, then we have enough operations to fully correlate the entire operator, and so this scaling ceases to be correct. We can provide a lower bound for the execution time in this case be setting $a = 2$, and so the time required scales with $O(2^{6\sqrt{2}n}) = N^{6\sqrt{2}}$, where N is the size of the space to be searched. In this case, our algorithm performs considerably worse than a classical search. However, if $a < 2$, the time required to perform this search will scale sub-linearly in N, and our algorithm will outperform classical search. If $a < 1$, our algorithm will outperform Grover's algorithm.

Note that the quantum circuit to form the oracle would actually need $2n_g$ gates to ensure that the final operator encodes a fixed number of solutions and so has bond size $O(1)$. However, this will only apply a prefactor to n^a in our scalings and will not affect our conclusions with respect to the performance of our algorithm at different a values. We also note that a quantum circuit to produce a specific oracle may not produce the maximum amount of correlation, and so this scaling will not apply. In this case, additional work would need to be performed to ascertain how large the MPO bonds become during computation of the oracle and thus whether Grover's algorithm would be superior to our classical MPO contraction algorithm.

6.3.1 Comparison to the work of Chamon and Mucciolo

Our results build upon earlier work by Chamon and Mucciolo (henceforth CM) [177]. In CM, a classical version of the MPS is defined, where all tensors are real and the the value obtained by contracting a MPS with a particular set of indices communicates the classical probability of measuring a result instead of the quantum mechanical coefficient. They then present an algorithm to determine the solution to a search problem given a classical logic circuit which produces the oracle. Our algorithm in section 6.3 is based upon this algorithm and so is very similar. However, as the algorithm of CM uses MPSs instead of MPOs, their are slight differences in the operation. In particular, instead of forming a MPO with the circuit, and then manipulating the indices of the MPO, they start with a particular probability distribution of input states represented by their classical MPS, and then apply the quantum circuit to each input state. By changing between a maximally uncertain input state and a more certain input state (one with a fixed value on a specific bit), they are able to determine a a single bit of a bit sequence that is a solution to the search problem. By repeating this process n times, where n is the number of bits, they can obtain a solution to the search problem.

Chamon and Mucciolo also perform a scaling analysis to ascertain in which cases their algorithm would outperform Grover's algorithm. Their results state that the maximum bond size in a MPS formed with n_g gates is $\min\left(2^{\lfloor\sqrt{2n_g}\rfloor}, 2^{\lfloor n/2\rfloor}\right)$, and so if $n_g = n^a$, the algorithm requires time of $O\left(n^{a+1} \times 2^{3\sqrt{2}n^{a/2}}\right)$. As Grover's Algorithm takes $O\left(2^{n/2}\right)$ time, the classical MPS algorithm presented can outperform Grover's algorithm if $a < 2$. Our methods are based upon theirs and so we obtain the same requirement for the value of a if a tensor network based classical search algorithm is to outperform Grover's algorithm. Their scaling is better than ours due to the exponentiation of a power of two rather than a power of four. However, our analysis of the final complexity of the MPO of an oracle provides the additional theoretical insight that a circuit to produce an oracle may create a large amount of entanglement, but must also destroy it before the computation concludes. This insight may be useful in the cases in which the circuit to produce an oracle is not maximally correlating.

6.4 Noisy simulations of Grover's algorithm

We have performed simulations of Grover's algorithm with fixed numbers of errors located between Grover iterations. This is in contrast to the usual noisy simulation approach, in which there is a probability of decoherence per time-step. In each case we can obtain a meaningful result by averaging over a statistically significant number of simulations with discrete errors occurring on pure states. However, the usual approach is directly connected to the theory of how a quantum computer would behave, where there is a fixed coherence time and so a set amount of decoherence per time unit. While our approach does not reflect this physical reality, it provides more straightforward insight into how errors will affect the simulation. We can also recover a probability of success given an error probability per time step from our results by computing the expected numbers of errors, and performing a weighted average over these different numbers. We will perform this calculation in section 6.4.5.

Existing simulations can simulate general quantum noise but are constrained to small numbers of qubits. Our simulations concern a regime where quantum computations are performed with a small but nonzero number of errors. Our error models are less general than other approaches, but we are able to simulate much larger systems by assuming a limited error model. We will obtain analytic results for the affect on one X, Z or Y error in sections 6.4.1, 6.4.1, and 6.4.3 respectively. We will also simulate larger numbers of errors in each of these sections.

All simulations were performed with 1000 stochastic noisy simulations for each number of qubits and number of errors before averaging. The resulting errors in the probability of success are the standard error. Simulations were performed for between 2 and 30 qubits and between 0 and 15 errors. In each stochastic run of a simulation, a single random solution to the search problem is selected.

In general, Grover's algorithm can have any number of solutions.To simplify our analytic calculations, we will only consider the case in which there is one solution to the search problem. We note that this is classically the most difficult case. As in the introduction to Grover's algorithm in section 2.2, we will use the notation that n is the number of qubits, $N = 2^n$ is the total number of elements in the search space, s is the solution, $|\psi\rangle = \frac{1}{\sqrt{N}} \sum_i |i\rangle$, $\beta = |s\rangle$ and $\alpha = \frac{1}{\sqrt{N-1}} \sum_{i \neq s} |i\rangle$. The state during a noiseless Grover search will be confined to the subspace spanned by $|\alpha\rangle$ and $|\beta\rangle$. Indeed, if we write a state in this subspace as $\cos\gamma|\alpha\rangle + \sin\gamma|\alpha\rangle$, the action of a Grover iteration can be written as a rotation:

$$\hat{G} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}, \tag{6.18}$$

where $\theta = 2\arccos\sqrt{1 - \frac{1}{N}}$.

6.4.1 X errors

Analytic result

We first consider the affect of a single X error on the success rate of Grover's algorithm. After k Grover iterations, the state of the system $|\phi\rangle$ will be:

$$\begin{aligned}|\phi\rangle = G^k|\psi\rangle &= \cos\left(\left(k+\frac{1}{2}\right)\theta\right)|\alpha\rangle + \sin\left(\left(k+\frac{1}{2}\right)\theta\right)|\beta\rangle \\ &= a(k)|\alpha\rangle + b(k)|\beta\rangle. \end{aligned} \tag{6.19}$$

We can rewrite this in terms of $|\psi\rangle$ and $|s\rangle$:

$$|\phi\rangle = a(k)\sqrt{\frac{N}{N-1}}|\psi\rangle + \left(b(k) - \frac{a(k)}{\sqrt{N-1}}\right)|s\rangle. \tag{6.20}$$

We now note that $|\psi\rangle$ is stabilised by a X gate, regardless of which qubit it acts upon. As such, when an X error occurs, we can consider its action only on the state $|s\rangle$. We write $X_i|s\rangle = |s_{X_i}\rangle$, which is a computational basis state which differs from the solution by only one bit.

To evolve this state under further Grover iterations, we must know how the states $|\psi\rangle$ and $|s_{X_i}\rangle$ evolve under Grover iterations:

$$G^gXG^k|\psi\rangle = a(k)\sqrt{\frac{N}{N_1}}G^g|\psi\rangle + \left(b(k) - \frac{a(k)}{\sqrt{N-1}}\right)G^g|s_{X_i}\rangle. \tag{6.21}$$

The evolution of $|\psi\rangle$ is straightforward, as it in the subspace spanned by $\{|\alpha\rangle, |\beta\rangle\}$ and is furthermore the initial state of the algorithm.

We can calculate the action of a Grover iteration (and subsequently many Grover iterations) on a single computational basis state which is not the solution by breaking the Grover iteration into the action of the oracle and the inversion about the mean. As the computational basis state we are considering is not the solution, the oracle acts on it as the identity. The action of a Grover iteration is then only given by the action of the inversion about the mean $(2|\psi\rangle\langle\psi| - I)$:

$$G|s_{X_i}\rangle = \frac{2}{\sqrt{N}}|\psi\rangle - |s_{X_i}\rangle \tag{6.22}$$

From (6.22) we can see that the action of a Grover iteration on a computational basis state which is not the solution is to apply a phase of -1 to the basis state and produce a small amount of $|\psi\rangle$. As such, each additional Grover iteration applied will produce another phase of -1 and an additional $\frac{2}{\sqrt{N}}|\psi\rangle$ (up to a phase of -1). By keeping track of the alternating phases, we can thus show that applying a number g of Grover iterations to the state $|s_{X_i}\rangle$ will produce the result

$$G^g|s_{X_i}\rangle = (-1)^{g \mod 2}|s_{X_i}\rangle + \frac{2}{\sqrt{N}}\sum_{j=0}^{g-1}(-1)^j\, G^{g-j-1}|\psi\rangle. \tag{6.23}$$

We are principally interested in the probability of obtaining the solution when we measure the system at the end of the algorithm. As such, we want to compute the coefficient $c_s = \langle s|G^g X G^k|\psi\rangle$. To compute this value, we must compute $\langle s|G^g|\psi\rangle$ and $\langle s|G^g|s_{X_i}\rangle$. We note that $\langle s|s_{X_i}\rangle = 0$ and $\langle s|G^g|\psi\rangle = \sin\left((g+1/2)\,\theta\right)$, and so:

$$\langle s|G^g|s_{X_i}\rangle = \frac{2}{N}\sum_{j=0}^{g-1}(-1)^j \sin\left(\left(g-j-1+\frac{1}{2}\right)\theta\right) \tag{6.24}$$

Explicitly computing this sum:

$$\langle s|G^g|s_{X_i}\rangle = \frac{1}{2}\sec\left(\frac{\theta}{2}\right)\left(\sin\left(g\theta\right)-\sin\left(g\pi\right)\right) \tag{6.25}$$

By using (6.25) and (6.21) and noting that $\sin(g\pi) = 0$, we obtain the coefficient of the solution given an error after k Grover iterations and with a total of $g-k$ Grover iterations:

$$\begin{aligned} c_s = \sqrt{\frac{N}{N-1}}\sin\left(\left(g+\frac{1}{2}\right)\theta\right)a(k) + \frac{1}{\sqrt{N}}\left(a(k)-\frac{b(k)}{\sqrt{N-1}}\right) \\ \sec\left(\frac{\theta}{2}\right)\sin\left(g\theta\right), \end{aligned} \tag{6.26}$$

Finally, we substitute the values of a, b and $\cos\left(\frac{\theta}{2}\right)$ into (6.26). We also set the total number of Grover iterations to $m = g + k$ to obtain:

$$\begin{aligned} c_s = &\sqrt{\frac{N}{N-1}} \sin\left(\left(m - k + \frac{1}{2}\right)\theta\right)\cos\left(\left(k + \frac{1}{2}\right)\theta\right) \\ &+ \frac{1}{\sqrt{N-1}}\left[\sin\left(\left(k+\frac{1}{2}\right)\theta\right) - \frac{1}{\sqrt{N-1}}\cos\left(\left(k+\frac{1}{2}\right)\theta\right)\right] \\ &\quad \sin\left((m-k)\theta\right) \end{aligned} \tag{6.27}$$

We can develop some intuition for (6.27) by noting that when an X error occurs, one of the computational basis states in $|\alpha\rangle$ is transformed into $|\beta\rangle$, and $|\beta\rangle$ is transformed into a computational basis state in $|\alpha\rangle$. Before the X error, the coefficient of $|\alpha\rangle$ will be $\sin\left((g + 1/2)\,\theta\right)$. After the X error, the coefficient of $|\beta\rangle$ will be effectively reset to that of $\frac{1}{\sqrt{N-1}}|\alpha\rangle$ (which is small) and so the largest contribution to the final coefficient of $|\beta\rangle$ will be the rotation of the $|\alpha\rangle$ state into the solution basis $|\beta\rangle$. The amount of rotation is given by the value $\cos\left(\left(m + \frac{1}{2}\right)\theta\right)$, and so together with the small change to the coefficient of $|\alpha\rangle$ brought about by the error, we recover the first term in (6.27). The second term in (6.27) is due to the smaller effect of the deviation from the Grover subspace given by $|s_{X_i}\rangle$ rotating towards the solution due to the Grover iteration (it is important to note that a component of $|s_{X_i}\rangle$ is in the Grover subspace as $|s_{X_i}\rangle$ is not orthogonal to $|\alpha\rangle$).

The value c_s is unphysical as it only corresponds to a single pure state discretisation with one error of a noisy Grover's algorithm evolution. As an error can occur at any time step, and will have the same effect no matter on which qubit it occurs, we can recover the probability of success by averaging over each possible value of k:

$$p_s = \frac{1}{m+1}\sum_{k=0}^{m} c_s^2 \tag{6.28}$$

With our exact expression (6.27), p_s will be very difficult to calculate. However, we can simplify (6.27) under certain assumptions.

As a first simplification, we can expand each trigonometric term involving m and set m to the optimal number of rotations in the noiseless case,

$m = \lfloor \pi/4\sqrt{N} \rfloor$. Doing so, we obtain:

$$c_s = \cos(k\theta)\left(\cos(k\theta) - \frac{1}{\sqrt{N-1}}\sin(k\theta)\right) + \frac{\cos\left(\left(k+\frac{1}{2}\right)\theta\right)}{\sqrt{N-1}} \left(\sin\left(\left(k+\frac{1}{2}\right)\theta\right) - \frac{1}{\sqrt{N-1}}\cos\left(\left(k+\frac{1}{2}\right)\theta\right)\right) \tag{6.29}$$

We may further simplify this expression by taking the limit of large N, in which case the only term in (6.29) which is relevant is the first one, and so $c_s = \cos(k\theta)^2$ and $p_s = \frac{1}{m+1}\sum_{k=0}^{m}\cos(k\theta)^4$, which gives the large N limit of p_s to be $3/8$.

We could extend our approach to obtain coefficients and probabilities of success for larger number of qubits. To do this calculation (for example) for two errors on different qubits, we would take the coefficient of $|\beta\rangle$ in (6.29) and the corresponding coefficient of $|\alpha\rangle$ as a base state, and then apply a similar analysis to the one employed above, where we express the state $|\phi\rangle$ in terms of $|\psi\rangle$ and $|s\rangle$. The basis state $|s\rangle$ will be moved to a different computational basis state, and so we will have two non-solution computational basis states slowly rotating towards the solution in the manner of (6.23). Computing such a probability will quickly become in-feasible for large numbers of errors.

Simulation results

We first consider the case in which only a single X error occurs. This will allow us to validate our simulation by comparing the results with the analytic model developed in the previous section. The probability of success for each number of qubits is shown in figure 6.3, as well as the analytic value calculated for the large N limit in the previous section of 0.375. As expected, we note that once the probability of success has converged, the number of qubits does not affect the probability of success (rather, it takes a fixed value). We see that the probability of success does converge quickly towards this value, and agrees with it from around 12 qubits onward. However, there are significant statistical variations around the analytic value.

The probabilities of success of simulations with several larger numbers of errors are shown in figure 6.4. We note that the probability of success for each number of errors converges towards a set value, and once it has converged the probability is independent of the number of qubits. For a

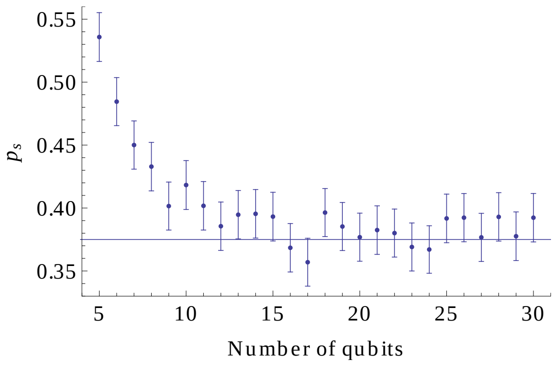


Figure 6.3: The probability of success of Grover's algorithm with a single X error. Also shown is the analytic value calculated for large N.

large enough number of qubits, it is thus the case that the probability of success is determined only by the number of errors, and not by the number of qubits. This is likely a result of structure of Grover's algorithm which does not depend on the number of qubits. That is, without errors the algorithm is restricted to a subspace of two quantum state, $|\alpha\rangle$ and $|\beta\rangle$, for any number of qubits. If errors do not occur on the same qubit, their affects interact very little (that there is little interaction between qubits in Grover's algorithm is indicated by the low MPO bond dimension of any number of Grover iterations) and have a similar affect on the proportion of the quantum state within the subspace $\{|\alpha\rangle, |\beta\rangle\}$ for any number of qubits. We thus suggest that the suitably large number of qubits we described above may be enough qubits that there is a small chance of two errors occurring on the same qubit. The requirement would be different for different numbers of errors, as we observe in figure 6.4.

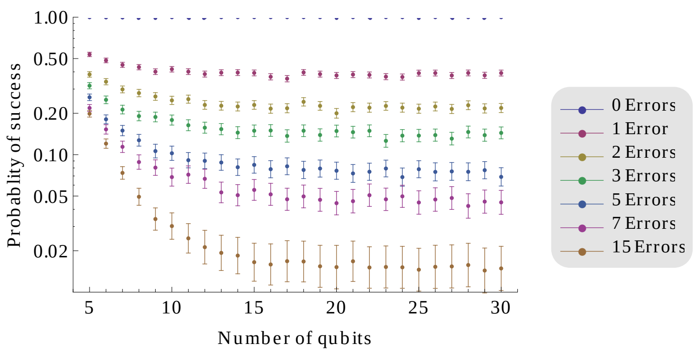


Figure 6.4: The probabilities of success of Grover's algorithm with several different numbers of X errors.

Given that the probability of success of Grover's algorithm depends only upon the number of errors that occur, the question naturally arises how this probability scales with the number of errors. We show the dependence of the probability of success on the number of errors for a 30 qubit simulation in figure 6.5. This the maximum number of qubits simulated, and so it should be the set of simulations that best meets the large N requirement. This data follows a linear relationship on the log-log plot shown, and so p_s follows a power law $p_s = an_e^b$, where n_e is the number of errors and b is the gradient of the linear fit on the plot. It appears that a linear relationship is present on the plot, but the first few data points do not follow it. As such, we have shown two linear fits on the plot, one in which all of the data (1 error to 15 errors inclusive) has been used to obtain a fit, and one in which the first tree data points have been disregarded. The latter fit, shown in blue, is clearly a better fit for the data.

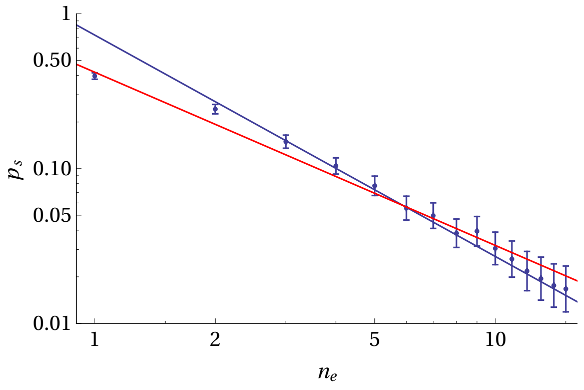


Figure 6.5: The probability of success of Grover's algorithm with several numbers of errors and 30 qubits, the maximum number simulated. Also shown are two linear fits to this data, including all of the data (red) and removing the first three points (blue).

It can be seen from figure 6.4 that from approximately 15 qubits onward, each of the p_s curves from 1 to 15 errors has converged to its final value. We can fit a linear relationship to the log-log decay of p_s with n_e for each number of qubits. The gradient of these fits is shown in figure 6.6. We can observe from this plot that the gradient of the fit is converged from approximately 15 qubits onward, as we would expect. An average of each of the converged gradients will give us a more accurate estimate of the power with which p_s decays with increasing numbers of errors. Averaging the gradients shown in figure 6.6 from 15 qubits to 30 qubits, we obtain $b = -1.409 \pm 0.009$.

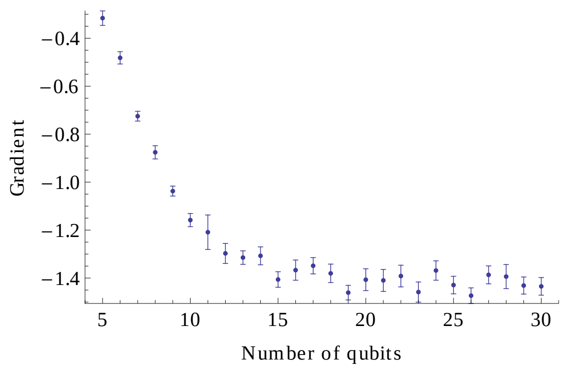


Figure 6.6: The gradient of the better of the linear fits shown in figure 6.5 for each number of qubits tested. This gradient gives the power in the decay of $p_s = a n_e^{-b}$, with n_e the number of errors.

6.4.2 Z errors

Analytic result

The second error model we consider is that of a fixed numbers of Z errors between Grover iterations. To calculate the final coefficient of the solution c_s and probability of success p_s in this case we will use a different approach to that we used with X errors. Instead, we will calculate the state of the system after an error has occurred, and then project this state back into the subspace formed by the orthogonal states $\{|\alpha\rangle, |\beta\rangle\}$.

We will first consider only one Z error as having occurred. In this case, after k Grover iterations, the system will be in the state (6.19):

$$\begin{aligned} |\phi\rangle = G^k|\psi\rangle &= \cos\left(\left(k+\frac{1}{2}\right)\theta\right)|\alpha\rangle + \sin\left(\left(k+\frac{1}{2}\right)\theta\right)|\beta\rangle \\ &= a(k)|\alpha\rangle + b(k)|\beta\rangle. \end{aligned} \tag{6.30}$$

After a Z gate, we will have a new state $Z|\phi\rangle$. We decompose this state into a part parallel and a part perpendicular to the Grover subspace:

$$Z_i|\phi\rangle = \langle\alpha|Z_i|\phi\rangle|\alpha\rangle + \langle\beta|Z_i|\phi\rangle|\beta\rangle + (Z_i|\phi\rangle)_\perp \tag{6.31}$$

We note that the Grover iteration is unitary, and so preserves inner products. Additionally, all states in the Grover subspace will remain in the Grover subspace under evolution by Grover iterations. As such, a state perpendicular to the Grover subspace will always be perpendicular to the Grover subspace if Grover iterations are the only kind of evolution that occurs.

After a Z gate is applied to one of the matrices, exactly half of the computational basis states will experience a phase shift of -1, and the other half will be unaffected. We will assume with no loss of generality that the basis state corresponding to the solution will have a 1 on the bit that the Z gate is applied to, and so will experience the phase shift. If the solution instead has a 0 in this location, we will obtain a result which is only different by a global phase. If the solution experiences a phase flip, then of the $N-1$ constituent states of $|\alpha\rangle$, $\frac{N}{2}-1$ will experience a phase flip and $\frac{N}{2}$ will not. As such, $\langle\alpha|Z_i|\phi\rangle = \frac{a}{N-1}$, and $\langle\beta|Z_i|\phi\rangle = -b$, and we know the projection of $Z_i|\phi\rangle$ into the Grover subspace (6.31). It is straightforward to write the part of $Z_i|\phi\rangle$ parallel to the Grover subspace parametrically:

$$(Z_i|\phi\rangle)_{||} = s\cos\gamma|\alpha\rangle + s\sin\gamma|\beta\rangle,$$
$$s = \sqrt{\frac{a(k)^2}{(N-1)^2} + b(k)^2}, \quad \gamma = \tan^{-1}\left(\frac{b(k)(N-1)}{a(k)}\right) \tag{6.32}$$

Each Grover iteration applies a phase of θ in the Grover subspace, so if we allow $m-k$ Grover iterations to happen after a Z error, making a total of m iterations, the part of the state in the Grover subspace will be $s\cos\left(\gamma + (m-k)\theta\right)|\alpha\rangle + s\sin\left(\gamma + (m-k)\theta\right)|\beta\rangle$. We are interested in the coefficient c_s of the solution basis state, which is given by $s\sin\left(\gamma + (m-k)\theta\right)|\beta\rangle$. Expanding this expression and substituting the values of a and b, we have:

$$\begin{aligned} c_s &= \sin\left((k+1/2)\theta\right)\cos((m-k)\theta) \\ &\quad + \frac{1}{N-1}\cos\left((k+1/2)\theta\right)\sin((m-k)\theta) \end{aligned} \tag{6.33}$$

As with the c_s expression for a single X error, we can expand each trigonometric term involving m and set $m = \lfloor\frac{\pi}{4}\sqrt{N}\rfloor$, and so $\cos\left(\left(m+\frac{1}{2}\right)\theta\right) = 0$ and $\sin\left(\left(m+\frac{1}{2}\right)\theta\right) = 0$. Doing so, we obtain a new expression for c_s:

$$c_s = \left[\sin\left(\left(k+\frac{1}{2}\right)\theta\right)\right]^2 + \frac{1}{N-1}\left[\cos\left(\left(k+\frac{1}{2}\right)\theta\right)\right]^2. \tag{6.34}$$

Taking the large N limit in (6.34) eliminates the second term. We can then square c_s and sum over the possible values of k to obtain the probability of success $p_s = \frac{1}{m+1}\sum_{k=0}^{m}\cos(k\theta)^4 = 3/8$. This is the same value as the large N limit of p_s for a single X error, which is due to the similarity between (6.29) and (6.34).

Simulation results

We follow similar analysis techniques for the results of simulations with Z errors as for those with X errors. In figure 6.7 we show p_s for between 5 and 30 qubits and select numbers of errors between 0 and 15. The expected

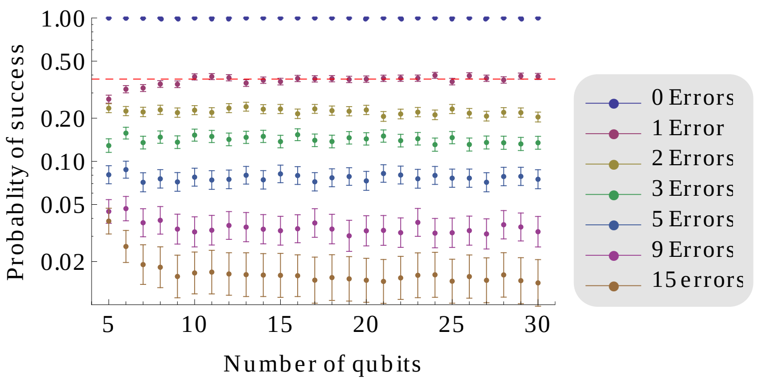


Figure 6.7: The probability of success of Grover's algorithm with several fixed numbers of Z errors and varying number of qubits. The expected value of p_s for 1 error in the large N limit is shown as a red dashed line.

value of p_s with 1 error in the large N limit is shown as a red dashed line. It is clear that the results agree with the expected value for one error from around 8 qubits onward. As with our X error results, we observe that each curve quickly converges to its final value, at which point only the number of errors is important to determine p_s. By comparison with figure 6.4 we see that the Z error curves converge more quickly than those for X errors. Our explanation of the large N behaviour at which the number of qubits does not matter in section 6.4.1 did not take account of the effect of two or more errors occurring on the same qubit. If the effect of this event is more extreme, the same explanation for the convergence of curves such as those in figure 6.7 may hold, but the convergence may occur more slowly. There is thus room in our explanation of large N behaviour for different convergence rates with different error models. The curves corresponding to each number of Z errors between 1 and 15 seem to be converged from approximately 9 qubits onward.

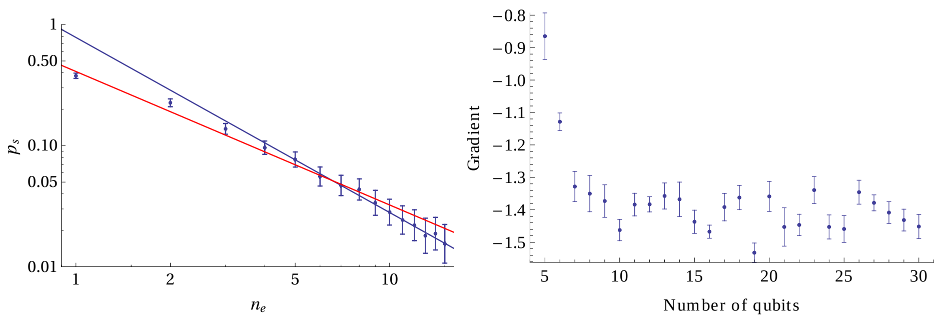


Figure 6.8: (a) The probability of success p_s of Grover's algorithm with 30 qubits and different numbers of errors, as well as two fits to the data, one (red) fitting to all data points and the other (blue) fitting to all but the first three of the data points. (b) The gradients of log-log fits to p_s against the number of errors for different numbers of qubits.

In figure 6.8a we show the dependence of p_s on the number of Z errors for 30 qubits. A linear relation is visible here for all but the first few points indicating a power law $p_s = ax^b$. The blue line in the figure is found with a linear fit excluding the first three data points, and provides a better fit than the red line, which is fitted to all data points. We show the gradients of linear log-log fits for each p_s, n_e curve, where n_e is the number of errors, for each number of qubits in figure 6.8b. The gradients converge to a roughly common value from around 9 qubits onward, as would be expected from the results shown in figure 6.7. Averaging over the gradients from 9 qubits onward provides us with the power of the decay $b = -1.41 \pm 0.01$. This value is the same within the standard error as that found in section 6.4.1 for X errors. This indicates that in the large N limit, both X and Z errors (each in isolation) have the same affect on the success of Grover's algorithm. Our analytic calculation of the affect of one X or one Z error found that in the large N limit p_s was found in one case by averaging $\cos(k\theta)^4$ over all k values, and in the other by averaging $\sin(k\theta)^4$ over all k values. These calculations produced the same result. It is likely that a similar result would follow with multiple errors, whereby averaging over large numbers of errors locations with two different functions would produce the same result.

The power of the decay of p_s with Z errors converges more quickly than that for X errors. This reflects the physical intuition that a X error swaps a state from the $|\alpha\rangle$ subspace and the solution state $|s\rangle$, but largely preserves the structure of the $\{|\alpha\rangle, |\beta\rangle\}$ subspace. Conversely, a Z error flips the sign of roughly half of the computational basis states in the $|\alpha\rangle$ state and so all but removes this state from the $\{|\alpha\rangle, |\beta\rangle\}$ subspace. A Z error is thus intuitively the worse error.

A comparison of figures 6.4 and 6.7 shows that as well as the power of the decay of p_s being the same for X and Z errors, p_s is similar for each number of errors in the large N limit. We illustrate this effect in figure 6.9 where we show the difference between p_s for ten X errors and each number of qubits, and p_s for ten Z errors and each corresponding number of qubits. This difference decreases exponentially between 5 and 15 qubits, and then remains roughly constant. There is a high amount of variation in this constant region and the difference takes both positive and negative values (we have taken the absolute value of the difference in figure 6.9) so this constant value is a result of the statistical noise visible in the variations of each converged point around its constant value in figure 6.4 and 6.7. In a simulation with a greater number of stochastic noisy pure state runs, this difference would be expected to plateau at a lower value.

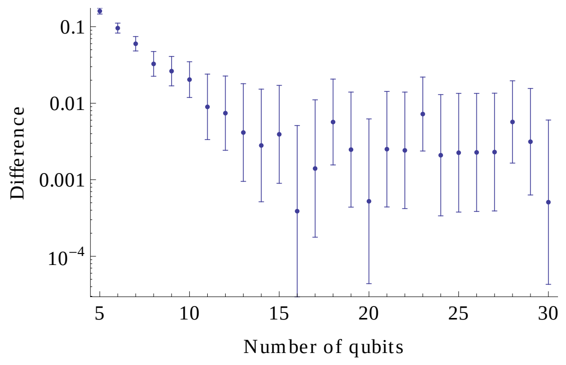


Figure 6.9: The absolute value of the difference between p_s for X errors and for Z errors with $n_e = 10$.

6.4.3 Y errors

Analytic result

The final error model we address analytically is that of having a small number of Y errors occur between Grover iterations. As with the X and Z errors, we focus first on the probability of success for Grover's algorithm with a single Y error. Allowing k iterations before an error occurs, the system will be in the state (6.20). As in the case of Z errors, we will determine the final result of a single Y error by projecting the resulting state $Y|\phi\rangle$ into the Grover subspace. To this end, we first consider the term $\langle\alpha|X_iZ_i|\psi\rangle$ which is equivalent to $\langle\alpha|Y_i|\psi\rangle$ up to a global phase. We assume with no loss of generality that the ith bit of the solution is a 1 and so Z_i applies a phase shift to the solution. As with our analysis of a single Z error, the case in which Z_i does not apply a phase shift to the solution will be different only by a global phase of -1. Here, $|\alpha\rangle$ will be made up of one more state with a 1_i than with 0_i. The Z error will apply a phase to half of the computational basis states in $|\psi\rangle$ (those with 1_i), and so after the application of X_i, all computational basis states with 0_i will have a phase of -1. Thus $\langle\alpha|X_iZ_i|\psi\rangle = -\frac{1}{\sqrt{N-1}}\frac{1}{\sqrt{N}}$. Additionally, as the solution has 1_i, $\langle\alpha|X_iZ_i|s\rangle = -\frac{1}{\sqrt{N-1}}$. We also note that $\langle s|X_iZ_i|\psi\rangle = \frac{1}{\sqrt{N-1}}$ and $\langle s|X_iZ_i|\psi\rangle = 0$. As such, we can write the state $X_iZ_i|\phi\rangle$ with components parallel and perpendicular to the Grover subspace

$$X_iZ_i|\phi\rangle = \frac{b(k)}{\sqrt{N-1}}|\alpha\rangle + \frac{a(k)}{\sqrt{N-1}}|s\rangle + (|\phi\rangle)_\perp \tag{6.35}$$

We can write the part of this vector parallel to the Grover subspace in parametric form:

$$\begin{aligned}(X_iZ_i|\phi\rangle)_\parallel &= s\cos(\gamma)|\alpha\rangle + s\sin(\gamma)|\beta\rangle, \quad s = \frac{1}{\sqrt{N-1}},\\ \gamma &= \tan^{-1}\left(\cot\left(\left(k+\frac{1}{2}\right)\theta\right)\right) = -\frac{\pi}{2} + \left(k-\frac{1}{2}\right)\theta. \end{aligned}\tag{6.36}$$

We apply evolution under $m-k$ additional Grover iterations to (6.36) to obtain the final component of the state parallel to the Grover subspace:

$$\begin{aligned}(X_iZ_i|\phi\rangle)_\parallel = \frac{1}{\sqrt{N-1}}\Bigg(&\cos\left(-\frac{\pi}{2} + \left(m-\frac{1}{2}\right)\theta\right)|\alpha\rangle\\ &+\sin\left(-\frac{\pi}{2} + \left(m-\frac{1}{2}\right)\theta\right)|\beta\rangle\Bigg).\end{aligned}\tag{6.37}$$

From (6.37) we can find the coefficient of the solution after m iterations with an error after k iterations:

$$c_s = -\frac{1}{\sqrt{N-1}}\cos\left(\left(m-\frac{1}{2}\right)\theta\right). \tag{6.38}$$

We note that (6.38) does not depend upon k and so the coefficient in the search state with one error is independent of when the error occurs, as long as it occurs between Grover iterations. Averaging c_s^2 over all possible k values will thus give the value $p_s = c_s^2 = \frac{1}{N-1} \cos\left(\left(m - \frac{1}{2}\right)\theta\right)^2$. This value has two immediate implications. Firstly, if we set m to the optimal noiseless value of $m = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$, then $\cos\left(\left(m - \frac{1}{2}\right)\theta\right) \approx 0$, and so $p_s \approx 0$ (this approximation becomes better as N becomes larger). Secondly, if we take the large N limit, p_s will approach zero no matter the value of m. This behaviour is very different to that calculated and observed in the previous sections for X and Z errors. Finally, we note that for small values of θ, $\cos(\pi/2 + \theta) = -\theta$, and so for large N and $m = \lfloor \frac{\pi}{4}\sqrt{N} \rfloor$, $p_s \approx \frac{1}{N^2} = 2^{-2n}$, and so we would expect to see an exponential decay of p_s for one Y error with increasing numbers of qubits.

Simulation results

In figure 6.10 we show the values of p_s obtained by simulating Grover's algorithm with one Y error. We also show the line of best fit through this

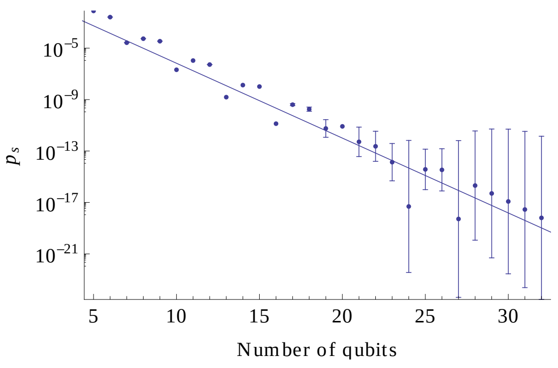


Figure 6.10: The probability of success of Grover's algorithm with one Y error, as well as a linear fit through $\log p_s$.

data, which gives an exponential decay $p_s = a2^{-bn}$ where $b = 1.93 \pm 0.09$. This is the same (within the standard error) as the expected decay rate of 2^{-2n}. The deviations of the values obtained from a monotonic exponential decrease are most likely due to the large N approximations made in our analysis not applying in cases with smaller numbers of qubits.

We also obtain an exponential decrease in p_s for other odd numbers of errors. The exponential decay of p_s with n for 15 errors in shown in figure 6.11a, as well as the line of best fit for the decay. For 15 errors, we

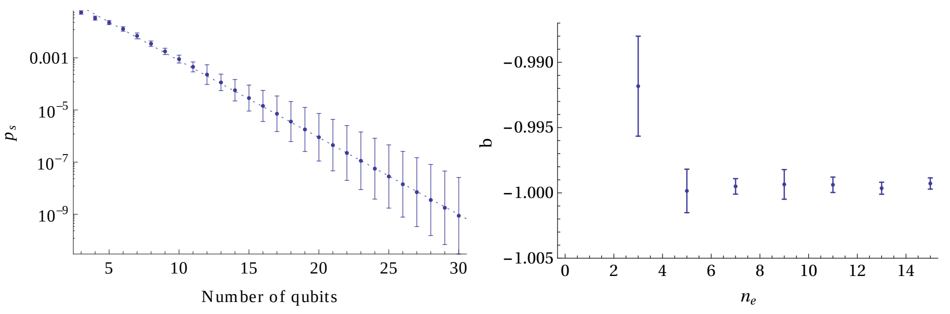


Figure 6.11: (a) The probability of success of Grover's algorithm with 15 Y errors as well as the line of best fit. (b) The gradients of fits to p_s versus n for different odd numbers of errors.

disregard the first three data points (which clearly do not fall on the line of best fit) and obtain a gradient of $b = -0.9959 \pm 0.0009$, which is also the decay rate in $p_s = a2^{bn}$. Given the closeness of this value to 1 and the fact that we are using numbers of qubits (5 and above) for which the large N approximation may not hold to a high degree of accuracy, it seems that for large N, $p_s = a2^{-n}$.

For even numbers of errors, we find two different regimes of behaviour. At low numbers of errors, the value of p_s decays inverse polynomially, while for large numbers of errors, p_s decays exponentially. We show examples of these two regimes in figure 6.12, where we display p_s for 2 errors and 14 errors. The first plot is a clear power law decay, but the second plot has a few points which are clearly different to the otherwise exponential decay. It is difficult to say whether these points are artifacts of using an insufficient number of stochastic simulation runs or whether they are an underlying feature of the physics. Nonetheless, these points illustrate the difficulty in assessing when one regime stops and the other starts. That is, numbers of errors between 2 and 14 generally seem to contain some points which lie on a straight line in a log plot, and others which lie on a straight line in a log-log plot. It is difficult to say which to what degree these points are statistical, or to draw further conclusions about the decay of p_s.

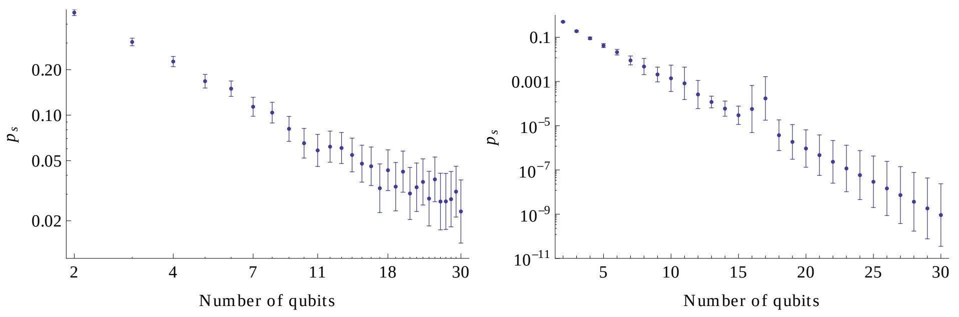


Figure 6.12: The probability of success of Grover's algorithm with two even numbers of errors, (a) 2 and (b) 14.

6.4.4 Depolarising noise

The final noise model we consider is that of depolarising noise. We simulate depolarising noise by applying a fixed number of errors as for our analyses of the affects of X, Y, and Z errors, but when an error occurs we randomly select which of X, Y or Z we apply. We expect the results to follow in part from the processes we found in the previous error analyses; however, including multiple kinds of error allows the different errors to interact and so additional effects will be present.

We observe a convergence of p_s towards a fixed value for each number of errors as the number of qubits is increased. This convergence is illustrated in figure 6.13. However, the rate of convergence is much slower than that observed in either X or Z errors. For example, p_s for 5 qubits seems to have converged from approximately 11 qubits onward. In line with this slower

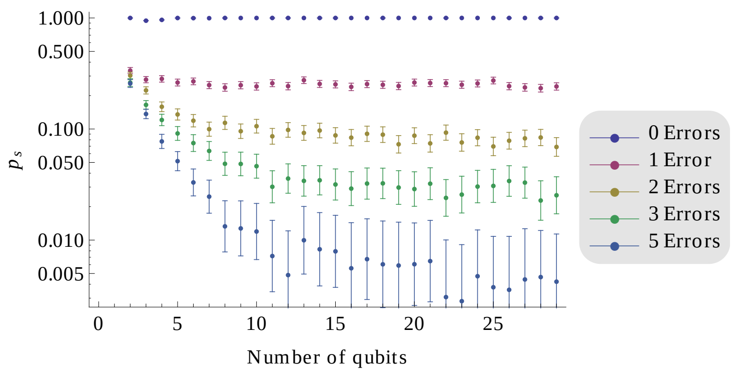


Figure 6.13: The probability of success of Grover's algorithm for numbers of errors between 0 and 5.

rate of convergence, we find that for the largest numbers of numbers of errors simulated, p_s does not appear to converge within the range of the numbers of qubits simulated. We illustrate this in figure 6.14 where we show p_s for 10 and 15 errors. The data for 10 errors appear to converge by approximately

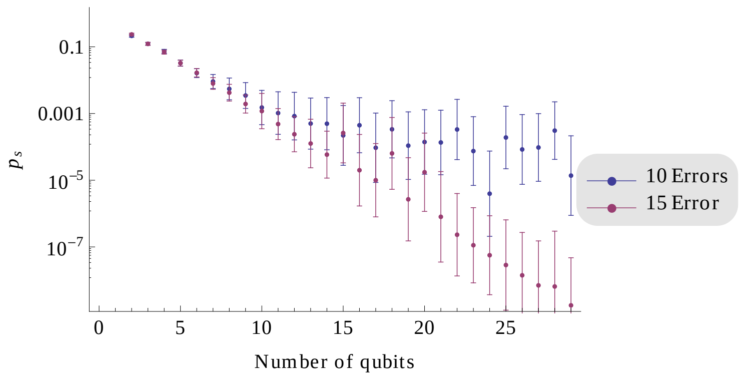


Figure 6.14: The probability of success of Grover's algorithm for 10 and 15 errors.

17 qubits. There are, however,several large temporary drops in p_s after this point. The large error in p_s at these points makes it difficult to determine if these drops are physical, or a result of insufficient numbers of stochastic simulation runs. The error would be reduced by simulating many more (for example 10000) stochastic pure state evolutions.

In contrast to the data for 10 errors, p_s does not seem to converge before 30 qubits, the maximum simulated. The value of p_s continues to decrease exponentially up to 30 qubits. Despite this, there are several points at which p_s temporarily increases by around an order of magnitude. As with 10 qubits, the large error in p_s at these points makes drawing any conclusions about their physicality or otherwise impossible. The large errors suggest that these points are statistical aberrations, and so more simulations should be performed to clarify them. Regardless of the presence of these points, the lack of convergence of the 15 qubit data means that we cannot include it when performing fits to the decay rate of p_s with increasing numbers of errors.

We qualitatively assessed that all numbers of errors between 0 and 11 converged within the range of numbers of qubits tested, and so considered only these data points wen performing fits. As with fits carried out in sections 6.4.1 and 6.4.2, the first three data points on each (n_e, p_s) curve did not appear to agree with the otherwise linear relationship on a log-log plot, and so we disregarded them. Using the remaining numbers of errors (between 3 and 11), we constructed linear fits on log-log plots for each number of qubits. The gradients of these fits, which is also the power of the decay of the probability of success, $p_s = an_e^b$, are shown in figure 6.15. The gradient of the fits increase until approximately 13 qubits. The red dashed line shows the line of best fit through this data, giving a rate of increase of n^c where $c = 1.93 \pm 0.07$ and n is the number of qubits. After 13 qubits the gradients appear to stabilise to their converged value of 4.51 ± 0.09 (this value is found by taking the mean of each gradient from 13 qubits onward). There is a single point which is much greater than this value at 27 qubits, where b takes the value $b = -12.435 \pm 3.14$. This unusually large value results from the presence of several anomalies which are likely statistical in nature, and are similar to those shown in figure 6.14. As with those points, further simulations could resolve whether this data point is an artifact of insufficient statistical sampling.

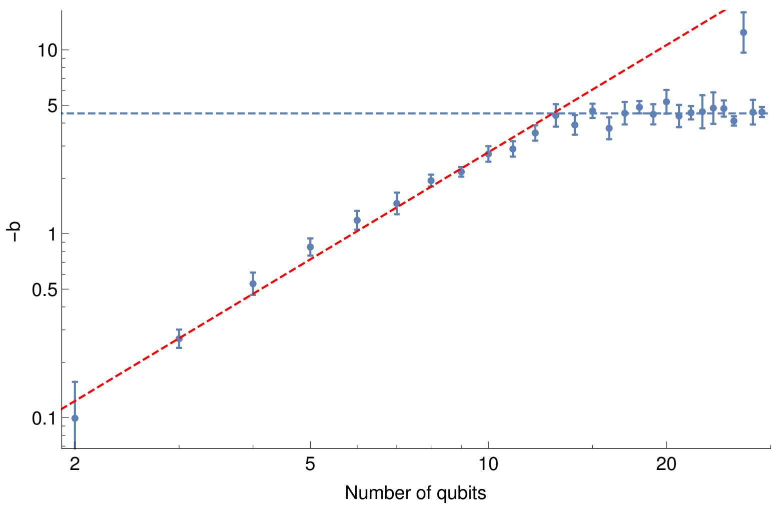


Figure 6.15: The gradients of linear fits on a log-log plot between the number of errors (for between 3 and 11 errors) and p_s for varying numbers of qubits.

6.4.5 Physical error models

While analysing the result of a fixed number of errors provides insight into the resilience of an algorithm, it is not a physical error model. A physical quantum computer would perform an algorithm as a series of gates. Each gate has a percentage change of failure in a given error model and noise on each qubit is discretised into a fixed probability of an error occurring at each time-step. We will extend our results to describe a small fixed chance of error on each qubit for each time-step.

To connect our results with a physical error model, we must describe how many errors occur under the physical error model. Assuming a fixed percentage of error each time step of ϵ, and a circuit involving n_t time-steps, the number of errors n_e will follow a binomial distribution. As we are interested in circuits with large numbers of qubits but only small numbers of errors, we take the limit that $e \ll 1$ or equivalently that $n_e \ll n_t$. In this case, the binomial distribution is very well approximated by a Poissonian distribution with mean ϵn_t:

$$p(n_e) = \frac{(\epsilon n_t)^{n_e}}{n_e!} e^{-\epsilon n_t}, \tag{6.39}$$

where $p(n_e)$ is the probability of a number of errors n_e occurring. Given this probability function for the number of errors, and a formula for the probability of success $p_s = a{n_e}^b$, we may find the expected value of the probability of success for a given n_t, ϵ:

$$\langle p_s \rangle = \sum_{n_e} p(n_e)\, p_s(n_e) = a e^{-\epsilon n_t} \sum_{n_e} \frac{(\epsilon n_t)^{n_e}}{n_e!} {n_e}^b \tag{6.40}$$

The decay rates for p_s which we found in the previous sections only provided a good fit for the data if we excluded points with three or less errors. As such, (6.40) will only provide a good approximation to p_s if the probability of obtaining $0, 1$ or 2 is very low. In addition, (6.40) is only valid if the number of qubits is much greater than the number of errors. While we have not derived quantitative bounds on how large the number of qubits must be, our results in the previous sections should provide some guidelines.

To obtain a relationship between p_s and the probability of error using our earlier results, we truncate the sum in (6.40) to terms with $n_e > 3$. For the other terms, we average p_s over all converged results for the specified number of errors and multiply by the corresponding Poisson distribution term $p(n_e)$. We show the expected value p_s for only X or Z errors (they are similar due to us using the approximation $p_s = ae^b$ over a large range of error numbers) in figure 6.16. As $\lambda = \epsilon n_t$ increases, the Poisson distribution narrows (it

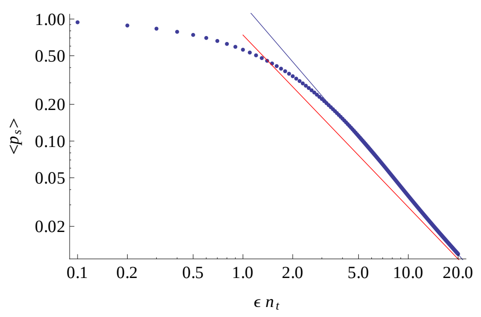


Figure 6.16: The expected probability of success of Grover's algorithm with an error model which has a fixed probability ϵ of a X error occurring per time step with n_t time steps. The blue line is a fit to the data between $\epsilon n_t = 10$ and $\epsilon n_t = 20$ while the red line is the expected long-term behaviour.

has a standard deviation of $\sqrt{\lambda}$) and so $\langle p_s \rangle$ approaches the expected value found if a δ distribution is used: $\langle p_s \rangle = a\lambda^b$. This relationship is shown in figure 6.16 with a red line, and it is evident that the data is approaching this line. We confirmed this approach by calculating $\langle p_s \rangle$ for much larger values of ϵn_t, but such a regime is not as relevant for our study on the effect of a limited number of errors. Also shown in figure 6.16 is the line of best fit from $\epsilon n_t = 10$ until $\epsilon n_t = 20$. The gradient of this line is -1.578 ± 0.002 and the line provides a very good fit to the data in the region considered. However, the data and this line of best fit diverge at higher values of ϵn_t. It is clear that if only a small region of ϵn_t is considered, spurious conclusions can be drawn about the large scale behaviour of $\langle p_s \rangle$.

A more dramatic illustration of the difference between short and long term behaviour of $\langle p_s \rangle$ can be found by consider depolarising noise. Including X, Y and Z errors, we show $\langle p_s \rangle$ in figure 6.17. We first show $\langle p_s \rangle$ for ϵn_t between 0 and 20 in figure 6.17a. The red line is is the curve $a(\epsilon n_t)^b$ and the blue line is the line of best fit between $\epsilon n_t = 10$ and $\epsilon n_t = 20$, which has a gradient of -6.293 ± 0.008. In this case, the blue line appears to provide a much better fit to the data. However, we also show $\langle p_s \rangle$ for ϵn_t up to 200, and it is clear that with large values of ϵn_t, $\langle p_s \rangle$ approaches $a(\epsilon n_t)^b$ to a very good approximation. We note that we have only performed direct simulations of Grover's algorithm with numbers of errors up to 15 and so extending our results to describe the performance of the algorithm with 200 errors involves a large amount of extrapolation. It is possible that the large N behaviour we analysed in sections 6.4.1, 6.4.2 and 6.4.4 persists,

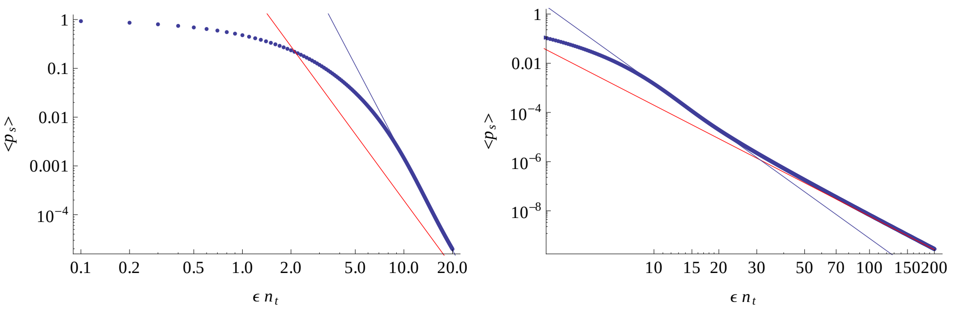


Figure 6.17: The expected probability of success of Grover's algorithm with an error model which has a fixed probability ϵ of a X, Y or Z error occurring per time step with n_t time steps. Two different ranges of ϵn_t are shown. The blue lines are a fit to the data between $\epsilon n_t = 10$ and $\epsilon n_t = 20$ while the red lines are the expected long-term behaviour.

but without further simulations of very large systems we would be unable to draw this conclusion definitively.

6.5 Discussion

In this chapter we have shown how to simulate Grover's algorithm with MPOs. The work of Saitoh [160] does include an example simulation of Grover's algorithm with a MPS. However, this example is for a specific case of the algorithm, and does not provide information about how to generalise the simulation. In this respect, our work generalises and extends this example. It is well known how to convert Grover's algorithm into a quantum circuit, and so a MPS simulation for any specific case of Grover's algorithm could be performed with the software package created by Saitoh [160]. However, we optimise the simulation of Grover's algorithm by using MPOs, and so a MPS simulation would not be as fast to perform. In particular, our work allows operators of the form G^a to be performed in $\log(a)$ time, while previous simulations would require a individual applications of the Grover iteration. Our simulations methods are also substantially faster than large-scale parallel state vector simulations.

The fact that the noiseless version of Grover's algorithm is restricted to a subspace of dimension 2 means that an instance of any size can be simulated with a two-dimensional vector and high precision complex numbers. However, our simulation techniques allow errors on any qubit to the be added to a large simulation of Grover's algorithm. We have considered X, Y and Z errors independently, as well as depolarising noise, but it would be possible to perform similar simulations with a variety of noise models.

Owing to the large number of error models used in the previous noisy simulations of Grover's algorithm, only the results of Salas [171] and Azuma [162, 163] concern error models which are similar to ours. Salas simulated Grover's algorithm with depolarising noise and finds an exponential damping law for the probability of success as the amount of noise increases. The derived scaling is of the probability of success at the first maximum of the algorithm, while we measure probability of success at the optimal number of noiseless iterations. With a physical error model, each iteration suffers decoherence and so the position of the maximum probability is shifted forward in iteration number. Our scalings and those of Salas are thus not directly comparable. Both results find an exponential dropoff of success probability with increasing error probability. Our results are more applicable to the operation of a quantum computer in which the level of noise is not well characterised.

Azuma perturbatively treats Grover's algorithm at low probabilities of Z errors occurring. In [162] this perturbation is calculated analytically up to third order, corresponding to three errors occurring. The work of Azuma thus extends the calculations we have performed in section 6.4.2, although our calculations are less involved. In [163], the perturbation theory is extended by a recurrence relation with CAS software up to the 39th order. The fact that these calculations have been performed suggests that similar calculations may be possible for other kinds of errors, such as X errors which we consider in section 6.4.1.

Many papers, including [171] find a scaling for the size of allowable noise in Grover's algorithm of N^{-a}. Our results suggest that for X, Y and Z errors as well as depolarising noise, the probability of success depends only on the number of errors occurring and not on the number of qubits, provided that the number of qubits is high enough. In this case the number of allowable errors remains the same regardless of N. With a physical error model including an error rate per time step per gate of ϵ and with n_g gates, the expected number of errors will be $n_g\epsilon$. The number of Grover iterations required is on the order of $N^{1/2}$, so if each Grover iteration requires n^b gates then $n_g = N^{1/2}n^b$, and for $n_g\epsilon$ to remain constant as N increases, we require that $\epsilon \sim N^{-1/2}n^{-b} = N^{-1/2}(\log_2(N))^{-b}$, which for limited ranges of N will appear to be a relation of the form $\epsilon \sim N^{-1/2}$. Salas finds that the permissible level of depolarising noise is $N^{-1.1}$ which does not agree with our conclusion. However, we note that range of the number of qubits in which the simulations of Salas operate are well below the level we find that give a constant probability of success for a reasonable number of errors regardless of the number of qubits. As such, we would expect the conclusions drawn in Salas to differ from those we draw, as we simulate Grover's algorithm for larger range of n. Performing more simulations at still greater values of n and with more stochastic runs per simulation would serve to increase our confidence in our conclusions and to tighten the bounds around some of the decay rates observed in sections 6.4.4.

Chapter 7

Conclusion

In this thesis, we have used MPOs and MPSs to simulate the QFT, Shor's algorithm and Grover's algorithm. To perform these simulations, original simulation techniques using MPOs were developed. As MPOs and MPSs allow efficient simulation of large systems with limited amounts of entanglement power and entanglement respectively, these data structures allow us to simulate large systems. Simulations of systems with more qubits than ours have been performed in the past for both Shor's algorithm and Grover's algorithm. However, it is worth noting that our simulations have taken place on a single core of a laptop computer, while the larger simulations summarised in chapters 5 and 6 took place on super computers with many cores. By moving to a more powerful computer, it would be possible to use our methods to simulate a considerably larger system.

Our simulations have been conducted with a limited numbers of quantum errors occurring. Such simulations are more computationally demanding than those occurring without errors, particularly in the case of Grover's algorithm in which all evolution in the error free case takes place in a Hilbert space of dimension 2, regardless of the number of qubits in the system. Our simulations of both Shor's algorithm and Grover's algorithm contain more qubits than earlier simulations incorporating noise. The ability to simulate large noisy quantum algorithms comes at the cost of inflexibility in the error model used. That is, we only simulate fixed numbers of single qubit X, Y and Z errors for Grover's algorithm, and only Z errors in Shor's algorithm. This limitation is inherent in the use of MPSs and MPOs, as these data structures can only efficiently simulate states with small amounts of entanglement. While the three Pauli errors are universal for single qubit noise, our simulations operate at a high level of abstraction and we apply errors only at limited locations to increase simulation speed. Simulations with more general noise models would have to use other techniques, in particular full density matrix calculations for fully general noise models.

By considering the operation of quantum algorithms through the lens of matrix product simulation, and by constructing MPOs, we have been able to make computational and theoretical points about the operation of the QFT, Shor's algorithm and Grover's algorithm. In chapter 3 we presented strong numerical evidence that the Schmidt coefficients across any bipartition of the MPO of the QFT do not increase as more qubits are added to the transform. This has the implication that a constant number of QFTs can be efficiently applied to a quantum state whose Schmidt rank grows only polynomially with the number of qubits. By reference to our results about the large Schmidt rank of MPSs in Shor's algorithm simulations with high r values in chapter 4, this suggests that the quantum speedup in Shor's algorithm depends upon the highly entangled (and thus high Schmidt rank) states generated by modular exponentiation. By considering the operation of Shor's algorithm with a MPS simulation, we have explained and generalised earlier results about the amount of entanglement present during error-free runs of Shor's algorithms. In Grover's algorithm, we found that the minimal bond rank of a MPO encoding an oracle with one solution is 2 (for any number of qubits) and the minimal rank for an oracle with an arbitrary number of solutions is at most $\sqrt{N}$ where N is the size of the search space. This result allow us to conclude that a circuit generating a Grover oracle may create entanglement during the operation of the circuit, but must destroy most of it before completion.

Our noisy simulations of Grover's algorithm and Shor's algorithm allowed us to derive scalings for the probability of success with the number of errors or the probability of an error in the regime where a small number of errors occurs. We found that the probability of success of Shor's algorithm scales with $e^{-a(N)n_e}$, where n_e is the number of Z errors occurring and $a(N)$ is a linear function of N, the number being factorised. This is an exponential decline in the success probability with increasing numbers of errors, and a super-exponential decrease in the success probability as the problem size increases. Such a scaling suggests that the algorithm is not viable for the factorisation of large numbers with even small numbers of errors. However, it must be noted that with quantum error-correction, a quantum computer could be made fault tolerant, and so no errors would occur. In this case, Shor's algorithm would present a viable method for factorising numbers, albeit with a large overhead in the numbers of qubits required to construct a fully fault-tolerant quantum computer.

We found that the probability of success of Grover's algorithm scales with an_e^b, where n_e is the number of errors occurring. This is a more favourable scaling than we observed in Shor's algorithm, although the probability of success of a single error-bound run varies greatly based upon which errors occur, where and when they occur. It is difficult to compare our results for either Grover's algorithm or Shor's algorithm to earlier results determining the effect of errors on algorithm success because of the disparate error models used in each earlier work and in this thesis. Where conflict does arise, our results are likely to apply to a broader regime of numbers of errors and qubits because of the increased size of our simulations relative to earlier noisy simulations of quantum algorithms.

The work presented in this thesis could be directly extended by proving that the Schmidt rank required of the MPO of the QFT for a particular precision does not increase with the number of qubits, by running similar simulations of larger systems on more powerful computers, or by incorporating different error models in simulations of Grover's algorithm. Running larger simulations would extend the results obtained for the scaling of the probability of success of the algorithms to larger numbers of qubits and reduce the error bounds in the scalings derived. Incorporating different error models would be informative if an error model could be derived that more accurately matched the noise likely to be encountered in a physical quantum computer. However, to maintain efficient simulability, any new noise model would still need to use a limited and discrete number of errors. Such an approach has limited utility at the lower levels of abstraction at which quantum computing architectures are considered. Owing to the large Schmidt rank encountered for simulations of Shor's algorithm with X errors, it is unlikely that other error models could be constructed which would permit efficient simulation with MPSs.

More broadly, there are many potential extensions of this work. The constant size of the MPO representation of the QFT per qubit as the number of qubits increases prompts us to ask if other favourable scalings follow. For example, can a relationship be found between the size of the MPO representation of an operator and the complexity of any potential circuit representation of the operator, and what implications would this have for circuit implementations of the QFT? Such a relationship exists for generation of MPS states [178], but one has not been found for MPOs. The ability of MPO simulations to simulate quantum algorithms with many qubits means that the determination of the entanglement of MPS and matrix product density operator states throughout the simulation would be worthwhile. We have used the log-negativity to do this in this thesis, but (relatively) quick calculation of other entanglement measures may be possible. It may also be possible to extend our techniques to perform circuit simulations with matrix product density operators, which would allow a more direct application of physically reasonable noise models. However, it is not clear that such simulations would be computationally efficient.

We have introduced quantum algorithm simulations incorporating MPOs by simulating the QFT, Shor's algorithm and Grover's algorithm, allowing us to perform large simulations and draw conclusions about the operation of, and effect of error on, these algorithms. It is likely that interesting results could be obtained by simulating other algorithms, particularly quantum simulation of small and simple systems. However, many algorithms could not be simulated using the methods developed in this thesis as they are too entangling or not easily represented with a small number of quantum gates and MPOs. We note that there are other quantum simulation techniques, such as stabilizer simulations, which permit the efficient simulation of other classes of quantum algorithm (notably quantum error correction). In this context, MPOs allow significant improvements in the speeds of some simulations and should be seen as another tool in the simulation arsenal of quantum algorithms research.

Bibliography

[1] Kieran J Woolfe, Charles D Hill, and Lloyd CL Hollenberg. Scale invariance and efficient classical simulation of the quantum fourier transform. *arXiv preprint arXiv:1406.0931*, 2014.

[2] PW Shor. Algorithms for Quantum Computation : Discrete Logarithms and Factoring. In *35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[3] P W Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[4] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6-7):467–488, 1982.

[5] LK Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 212–219, 1996.

[6] Lov K Grover. Quantum Mechanics Helps in Searching for a Needle in a Haystack. *Physical Review Letters*, 79(2):325–328, 1997.

[7] Daniel S Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Physical Review Letters*, 79(13):2586, 1997.

[8] Stephen Wiesner. Simulations of many-body quantum systems by a quantum computer. *arXiv preprint quant-ph/9603028*, 110, 1996.

[9] Gerardo Ortiz, JE Gubernatis, Emanuel Knill, and Raymond Laflamme. Quantum algorithms for fermionic simulations. *Physical Review A*, 64(2):022319, 2001.

[10] Dominic W Berry, Graeme Ahokas, Richard Cleve, and Barry C Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[11] Stephen P Jordan, Keith SM Lee, and John Preskill. Quantum algorithms for quantum field theories. *Science*, 336(6085):1130–1133, 2012.

[12] Alán Aspuru-Guzik, Anthony D Dutoi, Peter J Love, and Martin Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309(5741):1704–1707, 2005.

[13] Edward Farhi and Sam Gutmann. Quantum computation and decision trees. *Phys. Rev. A*, 58:915–928, Aug 1998.

[14] John Watrous. Quantum simulations of classical random walks and undirected graph connectivity. *Journal of Computer and System Sciences*, 62(2):376 – 391, 2001.

[15] Neil Shenvi, Julia Kempe, and K Birgitta Whaley. Quantum random-walk search algorithm. *Physical Review A*, 67(5):052307, 2003.

[16] Andrew M Childs and Jeffrey Goldstone. Spatial search by quantum walk. *Physical Review A*, 70(2):022314, 2004.

[17] Andris Ambainis, Julia Kempe, and Alexander Rivosh. Coins make quantum walks faster. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1099–1108. Society for Industrial and Applied Mathematics, 2005.

[18] Frédéric Magniez, Miklos Santha, and Mario Szegedy. Quantum algorithms for the triangle problem. *SIAM Journal on Computing*, 37(2):413–424, 2007.

[19] Harry Buhrman and Robert Špalek. Quantum verification of matrix products. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 880–889. ACM, 2006.

[20] Michele Mosca and Artur Ekert. The hidden subgroup problem and eigenvalue estimation on a quantum computer. In *Quantum Computing and Quantum Communications*, pages 174–188. Springer, 1999.

[21] Michelangelo Grigni, Leonard Schulman, Monica Vazirani, and Umesh Vazirani. Quantum mechanical algorithms for the nonabelian hidden subgroup problem. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 68–74. ACM, 2001.

[22] Richard Jozsa. Quantum factoring, discrete logarithms, and the hidden subgroup problem. *Computing in Science & Engineering*, 3(2):34–43, 2001.

[23] Mark Ettinger, Peter Høyer, and Emanuel Knill. The quantum query complexity of the hidden subgroup problem is polynomial. *Information Processing Letters*, 91(1):43–48, 2004.

[24] Greg Kuperberg. A subexponential-time quantum algorithm for the dihedral hidden subgroup problem. *SIAM Journal on Computing*, 35(1):170–188, 2005.

[25] Wim Van Dam, Sean Hallgren, and Lawrence Ip. Quantum algorithms for some hidden shift problems. *SIAM Journal on Computing*, 36(3):763–778, 2006.

[26] Andrew M Childs and Wim van Dam. Quantum algorithm for a generalized hidden shift problem. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1225–1232. Society for Industrial and Applied Mathematics, 2007.

[27] Andrew M Childs, Leonard J Schulman, and Umesh V Vazirani. Quantum algorithms for hidden nonlinear structures. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 395–404. IEEE, 2007.

[28] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, Joshua Lapan, Andrew Lundgren, and Daniel Preda. A quantum adiabatic evolution algorithm applied to random instances of an np-complete problem. *Science*, 292(5516):472–475, 2001.

[29] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[30] Jérémie Roland and Nicolas J Cerf. Quantum search by local adiabatic evolution. *Physical Review A*, 65(4):042308, 2002.

[31] Dorit Aharonov, Wim Van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM review*, 50(4):755–787, 2008.

[32] E Knill and R Laflamme. Power of One Bit of Quantum Information. *Phys. Rev. Lett.*, 81(25):5672–5675, December 1998.

[33] Andrew M. Childs and Wim van Dam. Quantum algorithms for algebraic problems. *Reviews of Modern Physics*, 82(1):1–52, 2010.

[34] I.M. Georgescu, S. Ashhab, and Franco Nori. Quantum simulation. *Reviews of Modern Physics*, 86(1):153–185, 2014.

[35] Michele Mosca. Quantum algorithms. In Robert A. Meyers, editor, *Computational Complexity*, pages 2303–2333. Springer New York, 2012.

[36] SalvadorElas Venegas-Andraca. Quantum walks: a comprehensive review. *Quantum Information Processing*, 11(5):1015–1106, 2012.

[37] S Jordan. Quantum algorithm zoo. `http://math.nist.gov/quantum/zoo/`, 2014. Accessed 1/2015.

[38] Noah Linden and Sandu Popescu. Good Dynamics versus Bad Kinematics: Is Entanglement Needed for Quantum Computation? *Phys. Rev. Lett.*, 87(4):47901, July 2001.

[39] R Jozsa and N Linden. On the role of entanglement in quantum-computational speed-up. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 459(2036):2011–2032, August 2003.

[40] Guifré Vidal. Efficient Classical Simulation of Slightly Entangled Quantum Computations. *Physical Review Letters*, 91(14):147902, 2003.

[41] IL Markov and Yaoyun Shi. Simulating quantum computation by contracting tensor networks. *SIAM Journal on Computing*, 38(3):963–981, 2008.

[42] Nadav Yoran. Efficiently contractable quantum circuits cannot produce much entanglement. *arXiv preprint arXiv:0802.1156*, pages 1–5, 2008.

[43] Román Orús and José I Latorre. Universality of entanglement and quantum-computation complexity. *Phys. Rev. A*, 69(5):52308, May 2004.

[44] Maarten Van den Nest, Akimasa Miyake, Wolfgang Dür, and Hans J Briegel. Universal Resources for Measurement-Based Quantum Computation. *Phys. Rev. Lett.*, 97(15):150504, October 2006.

[45] D Brußand C Macchiavello. Multipartite entanglement in quantum algorithms. *Phys. Rev. A*, 83:052313, 2011.

[46] Maarten Van den Nest. Universal Quantum Computation with Little Entanglement. *Physical Review Letters*, 110(6):060504, 2013.

[47] E. Knill, R. Laflamme, and W. H. Zurek. Resilient quantum computation: error models and thresholds. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 454(1969):365–384, 1998.

[48] Animesh Datta, Steven T Flammia, and Carlton M Caves. Entanglement and the power of one qubit. *Phys. Rev. A*, 72(4):42316, October 2005.

[49] Animesh Datta and Guifre Vidal. Role of entanglement and correlations in mixed-state quantum computation. *Physical Review A*, 75(4):042310, April 2007.

[50] Eli Biham, Gilles Brassard, Dan Kenigsberg, and Tal Mor. Quantum computing without entanglement. *Theoretical Computer Science*, 320(1):15–33, June 2004.

[51] Harold Ollivier and WH Zurek. Quantum discord: a measure of the quantumness of correlations. *Physical review letters*, 88(1):017901, 2001.

[52] L Henderson and V Vedral. Classical, quantum and total correlations. *Journal of Physics A: Mathematical and General*, 34(35):6899, 2001.

[53] Animesh Datta, Anil Shaji, and Carlton Caves. Quantum discord and the power of one qubit. *Phys. Rev. Lett.*, 100:050502, Feb 2008.

[54] Vlatko Vedral. The elusive source of quantum speedup. *Foundations of Physics*, 40(8):1141–1154, 2010.

[55] Aharon Brodutch and Daniel R. Terno. Entanglement, discord, and the power of quantum computation. *Phys. Rev. A*, 83:010301, Jan 2011.

[56] DA Lidar and TA Brun. *Quantum Error Correction.* Cambridge University Press, 2013.

[57] Austin G. Fowler, Matteo Mariantoni, John M. Martinis, and Andrew N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, September 2012.

[58] David S. Wang, Austin G. Fowler, and Lloyd C. L. Hollenberg. Surface code quantum computing with error rates over 1%. *Physical Review A*, 83(2):020302, 2011.

[59] Ashley M. Stephens. Fault-tolerant thresholds for quantum error correction with the surface code. *Physical Review A*, 89(2):022321, 2014.

[60] Steven R White. Density matrix formulation for quantum renormalization groups. *Physical Review Letters*, 69(19):2863–2866, 1992.

[61] U Schollwöck. The density-matrix renormalization group. *Reviews of Modern Physics*, 77(January):259–315, 2005.

[62] Stellan Östlund and Stefan Rommer. Thermodynamic limit of density matrix renormalization. *Phys. Rev. Lett.*, 75:3537–3540, Nov 1995.

[63] Jorge Dukelsky, Miguel A Martín-Delgado, Tomotoshi Nishino, and Germán Sierra. Equivalence of the variational matrix product method and the density matrix renormalization group applied to spin chains. *Europhysics Letters*, 43(4):457, 1998.

[64] F Verstraete and JI Cirac. Renormalization algorithms for quantum-many body systems in two and higher dimensions. *arXiv preprint cond-mat/0407066*, 2004.

[65] F Verstraete, JJ Garcia-Ripoll, and JI Cirac. Matrix product density operators: simulation of finite-temperature and dissipative systems. *Physical review letters*, 93:207204, 2004.

[66] Michael Zwolak and Guifré Vidal. Mixed-State Dynamics in One-Dimensional Quantum Lattice Systems: A Time-Dependent Superoperator Renormalization Algorithm. *Physical Review Letters*, 93:207205, 2004.

[67] B Pirvu, V Murg, J I Cirac, and F Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(2):025012, 2010.

[68] M. Bañuls, M. Hastings, F. Verstraete, and J. Cirac. Matrix Product States for Dynamical Simulation of Infinite Chains. *Physical Review Letters*, 102(24):240603, June 2009.

[69] Daniel Gottesman. The Heisenberg representation of quantum computers. *arXiv preprint quant-ph/9807006*, 1:1–20, 1998.

[70] Maarten Van den Nest. Classical simulation of quantum computation, the Gottesman-Knill theorem, and slightly beyond. *Quantum Information & Computation*, 10:0258–0271, 2010.

[71] Maarten Van Den Nest. Efficient classical simulations of quantum fourier transforms and normalizer circuits over abelian groups. *Quantum Information & Computation*, 13(11-12):1007–1037, 2013.

[72] Juan Bermejo-Vega and Maarten Van den Nest. Classical simulations of Abelian-group normalizer circuits with intermediate measurements. *Quantum Information & Computation*, 14(3):181–216, 2014.

[73] Leslie G Valiant. Quantum Computers that can be Simulated Classically in Polynomial Time. In *Proceedings of the ACM Symposium on the Theory of Computing*, page 114, 2001.

[74] R. Jozsa and a. Miyake. Matchgates and classical simulation of quantum circuits. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 464(2100):3089–3106, December 2008.

[75] E Knill. Fermionic linear optics and matchgates. *arXiv preprint quant-ph/0108033*, pages 1–17, 2001.

[76] Scott Aaronson and Alex Arkhipov. The computational complexity of linear optics. *Proceedings of the 43rd annual ACM Symposium on Theory of Computing*, pages 333–342, 2011.

[77] Nadav Yoran and AJ Short. Classical simulation of limited-width cluster-state quantum computation. *Physical Review Letters*, 96:170503, 2006.

[78] Richard Jozsa. On the simulation of quantum circuits. *arXiv preprint quant-ph/0603163*, pages 1–12, 2006.

[79] Martin Schwarz and Maarten Van den Nest. Simulating Quantum Circuits with Sparse Output Distributions. *arXiv preprint arXiv:1310.6749*, pages 1–21, 2013.

[80] Glosser.ca. Wikimedia commons. `http://commons.wikimedia.org/wiki/File:Bloch_Sphere.svg`. Accessed 15/12/2014.

[81] David Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[82] Adriano Barenco, Charles H Bennett, Richard Cleve, David P Divincenzo, Norman Margolus, Harald Weinfurter, Tycho Sleator, John A Smolin, and Peter Shor. Elementary gates for quantum computation. *Physical Review A*, 52(5):3457, 1995.

[83] E. Knill. Approximation by Quantum Circuits. *eprint arXiv:quant-ph/9508006*, August 1995.

[84] M. Boyer, G. Brassard, P. Høyer, and A. Tapp. Tight Bounds on Quantum Searching. *Fortschritte der Physik*, 46:493–505, 1998.

[85] MA Nielsen and IL Chuang. *Quantum computation and quantum information.* Cambridge University Press, Cambridge, 2000.

[86] Charles H Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. Strengths and weaknesses of quantum computing. *SIAM Journal on Computing*, 26(5):1510–1523, 1997.

[87] Christof Zalka. Fast versions of shor's quantum factoring algorithm. *arXiv preprint quant-ph/9806084*, 1998.

[88] Lov Grover. Fixed-Point Quantum Search. *Physical Review Letters*, 95(15):150501, 2005.

[89] Ari Mizel. Critically Damped Quantum Search. *Physical Review Letters*, 102(15):150501, April 2009.

[90] Richard Crandall and Carl Pomerance. *Prime numbers: a computational perspective.* New York, 2001.

[91] Austin G Fowler, Simon J Devitt, and Lloyd CL Hollenberg. Implementation of Shor's algorithm on a linear nearest neighbour qubit array. *Quantum Information & Computation*, 4(4):237–251, 2004.

[92] Stephane Beauregard. Circuit for Shor's algorithm using 2n+ 3 qubits. *Quantum Information & Computation*, 3(2):175–185, 2003.

[93] Yasuhiro Takahashi and Noboru Kunihiro. A fast quantum circuit for addition with few qubits. *Quantum Information & Computation*, 8(6&7):636–649, 2008.

[94] V Vedral, A Barenco, and A Ekert. Quantum networks for elementary arithmetic operations. *Physical review. A*, 54(1):147–153, 1996.

[95] SA Kutin. Shor's algorithm on a nearest-neighbor machine. *arXiv preprint quant-ph/0609001*, 2006.

[96] Rodney Van Meter and Kohei Itoh. Fast quantum modular exponentiation. *Physical Review A*, 71(5):052320, 2005.

[97] Yasuhiro Takahashi, Noboru Kunihiro, and Kazuo Ohta. The quantum fourier transform on a linear nearest neighbor architecture. *Quantum Information & Computation*, 7(4):383–391, 2007.

[98] Paul Pham and KM Svore. A 2D nearest-neighbor quantum architecture for factoring in polylogarithmic depth. *Quantum Information & Computation*, 13(11&12):937–962, 2013.

[99] Byung-Soo Choi and Rodney Van Meter. A $O(\sqrt{n})$-Depth Quantum Adder on the 2D NTC Quantum. *ACM Journal on Emerging Technologies in Computing Systems*, 8(3):1–22, 2012.

[100] Don Coppersmith. An approximate Fourier transform useful in quantum factoring. *arXiv preprint quant-ph/0201067*, 1994.

[101] Rb Griffiths and Cs Niu. Semiclassical Fourier transform for quantum computation. *Physical Review Letters*, 76(17):3228–3231, 1996.

[102] Artur K Ekert. Quantum cryptography based on Bell's theorem. *Phys. Rev. Lett.*, 67(6):661–663, August 1991.

[103] Charles H Bennett, Gilles Brassard, Claude Crepeau, Richard Jozsa, Asher Peres, and William K Wootters. Teleporting an unkown quantum state via dual classical and Eintein-Podolsky-Rosen channels. *Physical Review Letters*, 70(13):1895–1899, 1993.

[104] Charles H Bennett and Stephen J Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69:2881, 1992.

[105] R Horodecki and P Horodecki. Quantum entanglement. *Reviews of Modern Physics*, 81(2):865–942, 2009.

[106] Charles H Bennett, Herbert J Bernstein, Sandu Popescu, and Benjamin Schumacher. Concentrating partial entanglement by local operations. *Physical Review A*, 53(4):2046–2052, 1996.

[107] G. Vidal and R. Werner. Computable measure of entanglement. *Physical Review A*, 65(3):032314, 2002.

[108] M. B. Plenio. Logarithmic negativity: A full entanglement monotone that is not convex. *Phys. Rev. Lett.*, 95:090503, Aug 2005.

[109] Abner Shimony. Degree of entanglement. *Annals of the New York Academy of Sciences*, 755(1):675–679, 1995.

[110] Tzu-Chieh Wei and Paul Goldbart. Geometric measure of entanglement and applications to bipartite and multipartite quantum states. *Physical Review A*, 68(4):042307, October 2003.

[111] E. Knill. Resilient Quantum Computation. *Science*, 279(5349):342–345, January 1998.

[112] F. Verstraete, V. Murg, and J.I. Cirac. Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems. *Advances in Physics*, 57(2):143–224, 2008.

[113] Hitesh J. Changlani, Jesse M. Kinder, C. J. Umrigar, and Garnet Kin-Lic Chan. Approximating strongly correlated wave functions with correlator product states. *Physical Review B*, 80(24):245116, 2009.

[114] Y.-Y. Shi, L.-M. Duan, and G. Vidal. Classical simulation of quantum many-body systems with a tree tensor network. *Physical Review A*, 74(2):022320, 2006.

[115] G. Vidal. Entanglement Renormalization. *Physical Review Letters*, 99(22):220405, 2007.

[116] G. Evenbly and G. Vidal. Algorithms for entanglement renormalization. *Physical Review B*, 79(14):144108, 2009.

[117] Matteo Rizzi, Simone Montangero, and Guifre Vidal. Simulation of time evolution with multiscale entanglement renormalization ansatz. *Physical Review A*, 77(5):052328, 2008.

[118] D Perez-Garcia and F Verstraete. Matrix product state representations. *arXiv preprint quant-ph/ . . .*, pages 1–28, 2006.

[119] Norbert Schuch, Michael Wolf, Frank Verstraete, and J. Cirac. Computational Complexity of Projected Entangled Pair States. *Physical Review Letters*, 98(14):140506, 2007.

[120] Z. Y. Xie, J. Chen, M. P. Qin, J. W. Zhu, L. P. Yang, and T. Xiang. Coarse-graining renormalization by higher-order singular value decomposition. *Physical Review B*, 86(4):045139, 2012.

[121] Tomaž Prosen and Marko Žnidarič. Matrix product simulations of non-equilibrium steady states of quantum spin chains. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(02):P02035, 2009.

[122] F. Fröwis, V. Nebendahl, and W. Dür. Tensor operators : Constructions and applications for long-range interaction systems. *Physical Review A*, 81(6):062337, 2010.

[123] Michael Nielsen, Christopher Dawson, Jennifer Dodd, Alexei Gilchrist, Duncan Mortimer, Tobias Osborne, Michael Bremner, Aram Harrow, and Andrew Hines. Quantum dynamics as a physical resource. *Physical Review A*, 67(5):052301, May 2003.

[124] F. Verstraete, D. Porras, and J. Cirac. Density Matrix Renormalization Group and Periodic Boundary Conditions: A Quantum Information Perspective. *Physical Review Letters*, 93(22):227205, 2004.

[125] J. Eisert, M. Cramer, and M. B. Plenio. Colloquium: Area laws for the entanglement entropy. *Reviews of Modern Physics*, 82(1):277–306, 2010.

[126] Dorit Aharonov, Zeph Landau, and Johann Makowsky. The quantum FFT can be classsically simulated. *arXiv preprint quant-ph/0611156*, 2006.

[127] Nadav Yoran and Anthony Short. Efficient classical simulation of the approximate quantum Fourier transform. *Physical Review A*, 76(4):042321, 2007.

[128] Adriano Barenco, Artur Ekert, Kalle-Antti Suominen, and Pivi Törmä. Approximate quantum Fourier transform and decoherence. *Physical Review A*, 54:139, 1996.

[129] Austin Fowler and Lloyd Hollenberg. Scalability of Shors algorithm with a limited set of rotation gates. *Physical Review A*, 70(3):032329, 2004.

[130] Y. S. Nam and R. Blümel. Scaling laws for Shor's algorithm with a banded quantum Fourier transform. *Physical Review A*, 87(3):032333, 2013.

[131] Daniel E Browne. Efficient classical simulation of the quantum Fourier transform. *New Journal of Physics*, 9(5):146–146, 2007.

[132] Alastair A Abbott. De-quantisation of the quantum Fourier transform. *Applied Mathematics and Computation*, 219(1):3–13, 2012.

[133] J Tyson. Operator-Schmidt decomposition of the quantum Fourier transform on Bbb CN1 $\otimes$ Bbb CN2. *Journal of Physics A: Mathematical and General*, 6813:0–7, 2003.

[134] K M Obenland and A M Despain. A parallel quantum computer simulator.

[135] J Niwa, K Matsumoto, and H Imai. General-purpose parallel simulator for quantum computing. *Physical Review A*, 66:062317, 2002.

[136] K. De Raedt, K. Michielsen, H. De Raedt, B. Trieu, G. Arnold, M. Richter, Th. Lippert, H. Watanabe, and N. Ito. Massively parallel quantum computer simulator. *Computer Physics Communications*, 176(2):121–136, 2007.

[137] F Tabakin and B Juliá-Díaz. QCMPI: a parallel environment for quantum computing. *Comput. Phys. Commun.*, 180:948–964, 2009.

[138] Y. S. Nam and R. Blümel. Streamlining Shor's algorithm for potential hardware savings. *Physical Review A*, 87(6):060304, 2013.

[139] Y S Nam and R. Blümel. Robustness and performance scaling of a quantum computer with respect to a class of static defects. *Physical Review A*, 88:062310, 2013.

[140] David S Wang, Charles D Hill, and Lloyd CL Hollenberg. Simulations of shor's algorithm using matrix product states. *arXiv preprint arXiv:1501.07644*, 2015.

[141] S Parker and MB Plenio. Efficient factorization with a single pure qubit and $logN$ mixed qubits. *arXiv preprint quant-ph/0001066*, 85(14):3049–52, 2000.

[142] Vivien M Kendon and William J Munro. Entanglement and its role in Shor's algorithm. *Quantum Information & Computation*, 6(7):630–640, 2006.

[143] Yishai Shimoni, Daniel Shapira, and Ofer Biham. Entangled quantum states generated by Shors factoring algorithm. *Physical Review A*, 72(6):062308, 2005.

[144] Yonatan Most, Yishai Shimoni, and Ofer Biham. Entanglement of periodic states, the quantum Fourier transform, and Shors factoring algorithm. *Physical Review A*, 81(5):052306, 2010.

[145] C Miquel, Jp Paz, and R Perazzo. Factoring in a dissipative quantum computer. *Physical Review Aiew A*, 54(4):2605–2613, 1996.

[146] César Miquel, Juan Paz, and Wojciech Zurek. Quantum Computation with Phase Drift Errors. *Physical Review Letters*, 78(20):3971–3974, 1997.

[147] S. Parker and MB Plenio. Entanglement simulations of Shor's algorithm. *Journal of Modern Optics*, 49(8):1325–1353, 2002.

[148] L. Wei, Xiao Li, Xuedong Hu, and Franco Nori. Effects of dynamical phases in Shors factoring algorithm with operational delays. *Physical Review A*, 71(2):022317, 2005.

[149] Simon J Devitt, Austin G Fowler, and Lloyd CL Hollenberg. Robustness of Shor's algorithm. *Quantum Information & Computation*, 6(7):616–629, 2006.

[150] Ignacio García-Mata, Klaus Frahm, and Dima Shepelyansky. Effects of imperfections for Shors factorization algorithm. *Physical Review A*, 75(5):052311, 2007.

[151] Ignacio García-Mata, Klaus Frahm, and Dima Shepelyansky. Shors factorization algorithm with a single control qubit and imperfections. *Physical Review A*, 78(6):062323, 2008.

[152] Y S Nam and R. Blümel. Performance scaling of Shor s algorithm with a banded quantum Fourier transform. *Physical Review A*, 84:044303, 2012.

[153] Simona Caraiman and Vasile Manta. Parallel simulation of quantum search. *International Journal of Computers Communications & Control*, 5(5):634–641, 2010.

[154] Xiangwen Lu, Jiabin Yuan, and Weiwei Zhang. Workflow of the Grover algorithm simulation incorporating CUDA and GPGPU. *Computer Physics Communications*, 184(9):2035–2041, 2013.

[155] GF Viamontes, IL Markov, and JP Hayes. Improving gate-level simulation of quantum circuits. *Quantum Information Processing*, 2(5), 2003.

[156] GF Viamontes, Manoj Rajagopalan, IL Markov, and JP Hayes. Gate-level simulation of quantum circuits. In *Proceedings of the 2003 Asia and South Pacific Design Automation Conference*, pages 295–301, 2003.

[157] GF Viamontes, IL Markov, and JP Hayes. *Quantum Circuit Simulation*. Springer Netherlands, Dordrecht, 2009.

[158] A Kawaguchi, K Shimizu, Y Tokura, and N Imoto. Classical simulation of quantum algorithms using the tensor product representation. *arXiv preprint quant-ph/ . . .*, pages 2–5, 2004.

[159] Akira SaiToh and Masahiro Kitagawa. Matrix-product-state simulation of an extended Brüschweiler bulk-ensemble database search. *Physical Review A*, 73(6):062332, 2006.

[160] Akira SaiToh. ZKCM: A C++ library for multiprecision matrix computation with applications in quantum information. *Computer Physics Communications*, 184(8):2005–2020, August 2013.

[161] RJC Spreeuw and TW Hijmans. Robust quantum searching with spontaneously decaying qubits. *Physical Review A*, 76(2):022306, 2007.

[162] Hiroo Azuma. Decoherence in Grover's quantum algorithm: Perturbative approach. *Physical Review A*, 65(4):042311, April 2002.

[163] Hiroo Azuma. Higher-order perturbation theory for decoherence in Grovers algorithm. *Physical Review A*, 72(4):042305, October 2005.

[164] GL Long, YS Li, WL Zhang, and CC Tu. Dominant gate imperfection in Grover's quantum search algorithm. *Physical Review A*, 61:042305, 2000.

[165] Pil H. Song and ILki Kim. Computational leakage: Grovers algorithm with imperfections. *The European Physical Journal D - Atomic, Molecular and Optical Physics*, 23(2):299–303, May 2003.

[166] Daniel Shapira, Shay Mozes, and Ofer Biham. Effect of unitary noise on Grovers quantum search algorithm. *Physical Review A*, 67(4):042301, April 2003.

[167] B Pablo-Norman and M Ruiz-Altaba. Noise in Grover's quantum search algorithm. *Physical Review A*, 61:012301, 1999.

[168] Piotr Gawron, Jerzy Klamka, and Ryszard Winiarczyk. Noise effects in the quantum search algorithm from the viewpoint of computational complexity. *International Journal of Applied Mathematics and Computer Science*, 22(2):493–499, January 2012.

[169] Neil Shenvi, Kenneth Brown, and K. Whaley. Effects of a random noisy oracle on search algorithm complexity. *Physical Review A*, 68(5):052313, November 2003.

[170] O. V. Zhirov and D. L. Shepelyansky. Dissipative decoherence in the Grover algorithm. *The European Physical Journal D*, 38(2):405–408, March 2006.

[171] P. J. Salas. Noise effect on Grover algorithm. *The European Physical Journal D*, 46(2):365–373, October 2007.

[172] A. A. Pomeransky, O. V. Zhirov, and D. L. Shepelyansky. Phase diagram for the Grover algorithm with static imperfections. *The European Physical Journal D*, 31(1):131–135, 2004.

[173] Jeff P Barnes and Warren S Warren. Decoherence and programmable quantum computation. *Physical Review A*, 60(6):4363–4374, 1999.

[174] A. Romanelli, A. Auyuanet, and R. Donangelo. Quantum search algorithm as an open system. *Physica A: Statistical Mechanics and its Applications*, 375(1):133–139, 2007.

[175] T Gorin, L Lara, and G V López. Simulation of static and random errors on Grovers search algorithm implemented in an Ising nuclear spin chain quantum computer with a few qubits. *Journal of Physics B: Atomic, Molecular and Optical Physics*, 43(8):085508, 2010.

[176] Gregory Crosswhite and Dave Bacon. Finite automata for caching in matrix product algorithms. *Physical Review A*, 78(1):012356, 2008.

[177] Claudio Chamon and Eduardo R. Mucciolo. Virtual Parallel Computing and a Search Algorithm Using Matrix Product States. *Physical Review Letters*, 109(3):030503, July 2012.

[178] C. Schön, E. Solano, F. Verstraete, JI Cirac, and MM Wolf. Sequential generation of entangled multiqubit states. *Physical review letters*, 95(11):110503, 2005.